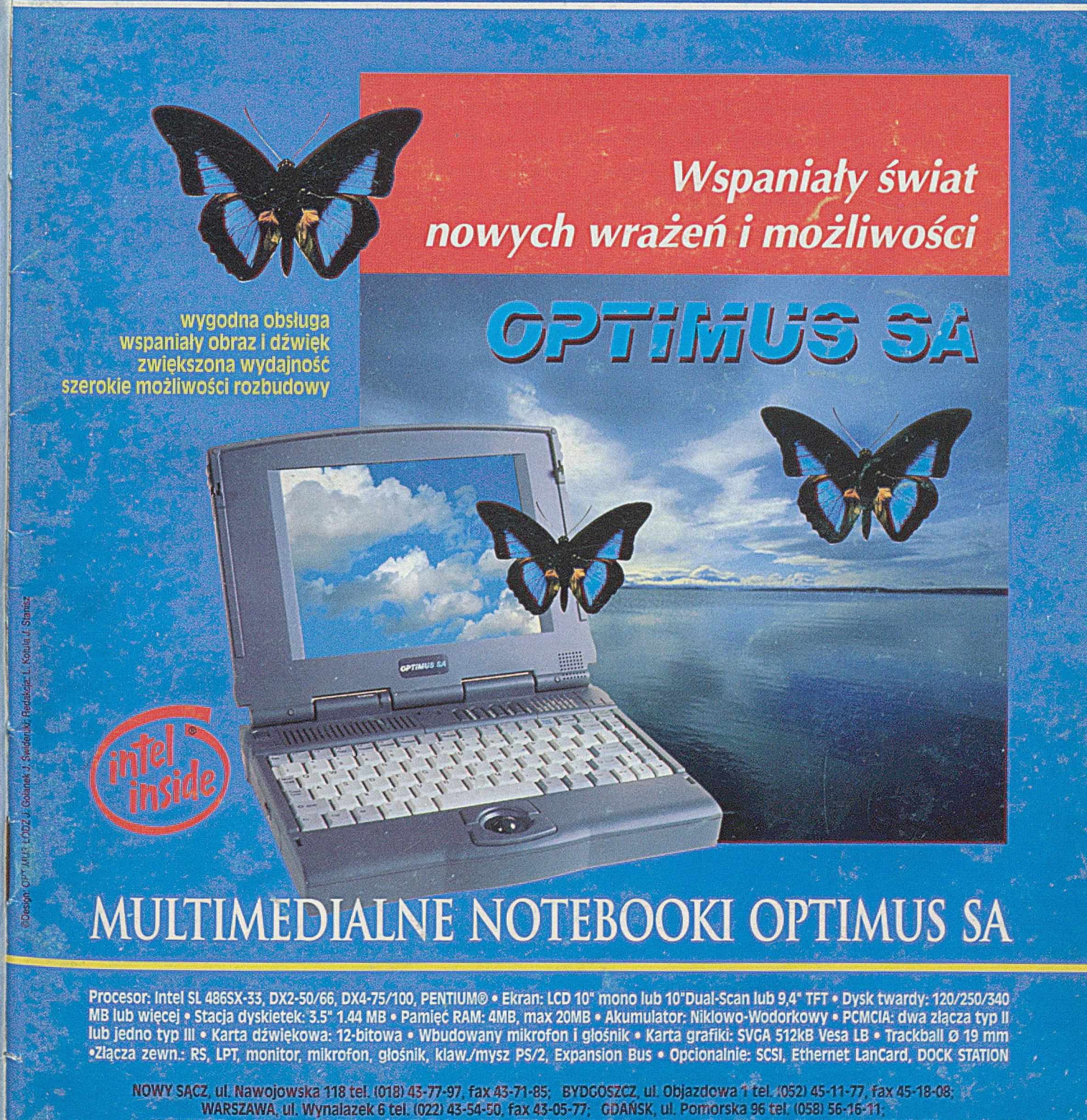
12
1994

WYDAWNICTWO SIGMA NOT

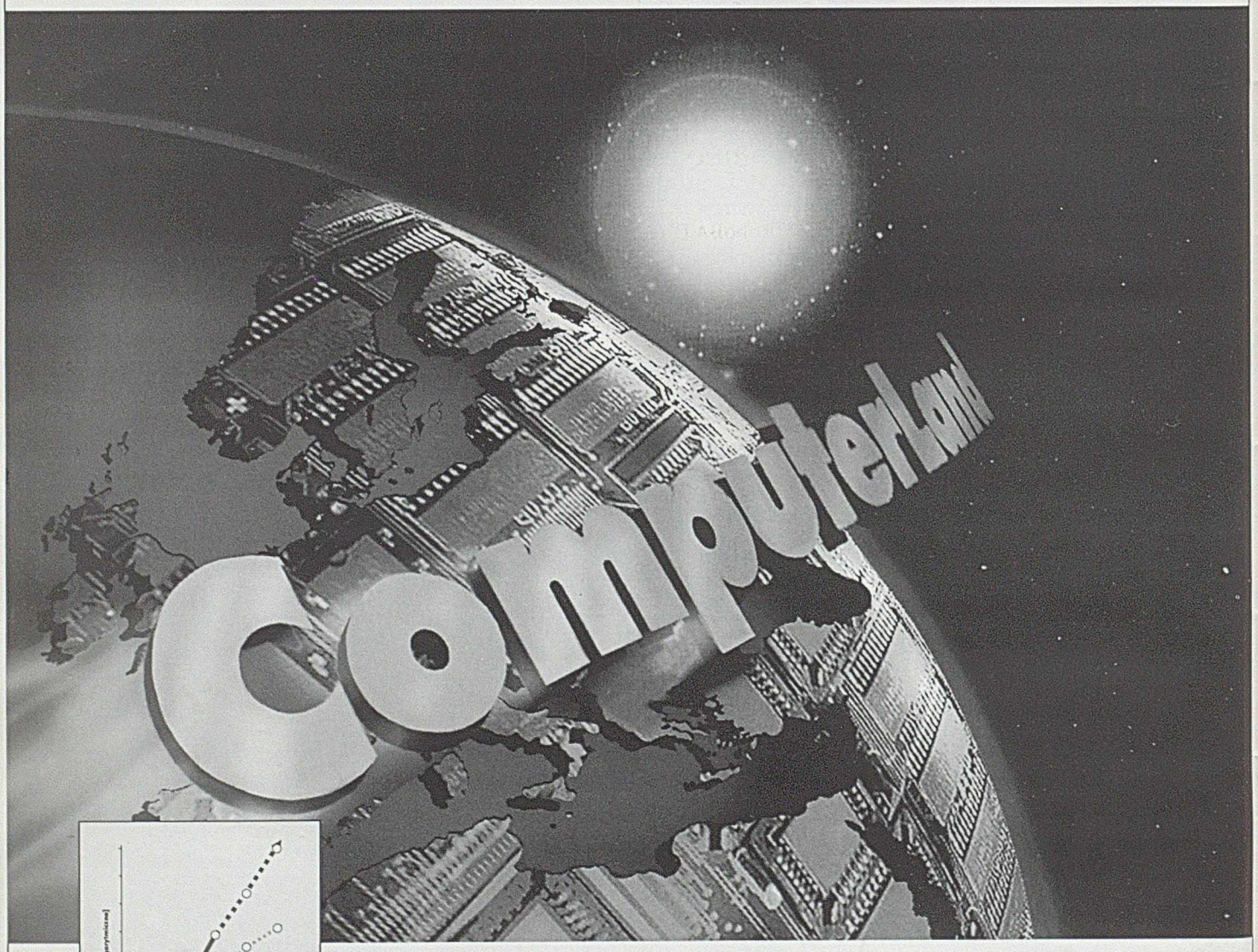
ISSN 0542-9951

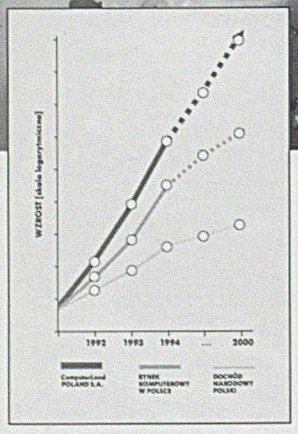
informatyka

P.1877/94


Wspaniały świat
nowych wrażeń i możliwości
OPTIMUS SA
wygodna obsługa
wspaniały obraz i dźwięk
zwiększona wydajność
szerokie możliwości rozbudowy
intel inside
MULTIMEDIALNE NOTEBOOKI OPTIMUS SA
Procesor: Intel SL 486SX-33, DX2-50/66, DX4-75/100, PENTIUM® • Ekran: LCD 10" mono lub 10"Dual-Scan lub 9,4" TFT • Dysk twardy: 120/250/340 MB lub więcej • Stacja dyskietek: 3.5" 1.44 MB • Pamięć RAM: 4MB, max 20MB • Akumulator: Niklowo-Wodorkowy • PCMCIA: dwa złącza typ II lub jedno typ III • Karta dźwiękowa: 12-bitowa • Wbudowany mikrofon i głośnik • Karta grafiki: SVGA 512kB Vesa LB • Trackball Ø 19 mm •Złącza zewn.: RS, LPT, monitor, mikrofon, głośnik, klaw./mysz PS/2, Expansion Bus • Opcionalnie: SCSI, Ethernet LanCard, DOCK STATION
NOWY SĄCZ, ul. Nawojowska 118 tel. (018) 43-77-97, fax 43-71-85; BYDGOSZCZ, ul. Objazdowa 1 tel. (052) 45-11-77, fax 45-18-08;
WARSZAWA, ul. Wynalazek 6 tel. (022) 43-54-50, fax 43-05-77; GDAŃSK, ul. Pomorska 96 tel. (058) 56-16-11;
©Design: OPTIMUS LODZ J. Golanek J. Świderski Redakcja: L. Kotula J. Stanisz

CHCEMY BYĆ CZĘŚCIĄ TWOJEGO SUKCESU





ComputerLand®

THE
INFORMATION
NETWORK

NAJWIĘKSZY INTEGRATOR SIECIOWYCH SYSTEMÓW KOMPUTEROWYCH W POLSCE

ComputerLand jest kluczowym polskim partnerem dla najlepszych firm świata:
AT&T, COMPAQ, Hewlett Packard, IBM, Microsoft, Novell.

Gdańsk	Kraków	Poznań	Szczecin	Warszawa	Wrocław
tel. (58) 37 44 45	tel. (12) 37 73 75	tel. (61) 517 100	tel. (91) 22 64 75	tel. (22) 20 52 35	tel. (71) 72 30 47
fax: (58) 37 44 47	fax: (12) 37 74 66	fax: (61) 517 102	fax: (91) 22 19 80	fax: (22) 20 52 36	fax: (71) 44 51 33

informatyka

Miesięcznik
ISSN 0542-9951
INDEKS 36124

nr 12 1994
Grudzień
Rok wydania XXX

P.1877/94

W numerze:

	Strona
PESEL – zbiór podstawowy – z Dariuszem Kupieckim, dyrektorem Rządowego Centrum Informatycznego PESEL rozmawia *Krystyna Karwicka*	2
I Kongres Informatyki Polskiej – Stan świadomości (kar)	3
POSTGRES jako narzędzie budowy aktywnych i dedukcyjnych systemów baz danych – *Marcin Januchta, Zbyszko Królikowski*	4
Próba naśladowania natury – algorytmy genetyczne w informatyce – *Halina Kwaśnicka*	12
Jakość oprogramowania – *Jerzy Auksztol*	17
AWEK – akustyczne wejście do komputera – *Czesław Basztura*	20
Systemy czasu rzeczywistego	
System operacyjny QNX jako narzędzie tworzenia aplikacji czasu rzeczywistego – *Jędrzej Ulasiewicz, Zbigniew Fryźlewicz*	22
Z kraju	
Softarg'94 – *Władysław Klepacz*	28
Spis treści rocznika	I-IV wewn. nr.

W najbliższych numerach – wyjątkowo na s. 16

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
prof. dr hab. Jan GOLIŃSKI
mgr Teresa JABŁOŃSKA (sekretarz redakcji)
mgr Krystyna KARWICKA-RYCHLEWICZ
Władysław KLEPACZ (redaktor naczelny)
mgr inż. Jan RYŻKO
dr Witold STANISZKIS
dr hab. Kazimierz SUBIETA
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel., fax: 39-14-34

Materiałów nie zamówionych redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Mazowiecka 12 telefon: 27-43-66 telefaks: 26-80-16

Miłych i spokojnych Świąt wszystkim Czytelnikom, Autorom i Współpracownikom

życzy Redakcja

Warunki prenumeraty na 1995 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-716 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573-139-11

Wpłaty na prenumeratę przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

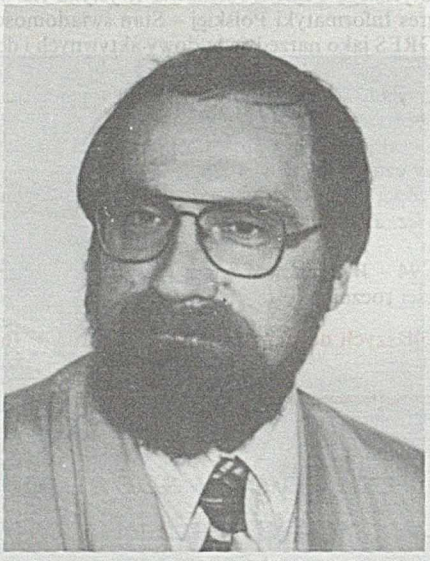
Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 45 000 zł, ulgowa 31 500 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 135 000, półroczna 270 000, za trzy kwartały 404 000, roczna 540 000
Ulgowa: kwartalna 94 500, półroczna 189 000, za trzy kwartały 283 500, roczna 378 000

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 433/94

PESEL zbiór podstawowy

Z Dariuszem Kupieckim dyrektorem Rządowego Centrum Informatycznego PESEL rozmawia Krystyna Karwicka



– PESEL był zawsze okryty tajemnicą. Dziennikarzom odmawiano wywiadów i praktycznie jakichkolwiek informacji na temat jego funkcjonowania. Czemu jest tak strzeżony?

– Dawniej rzeczywiście nie można było rozmawiać na temat PESEL-u, ale to nie wynikało, moim zdaniem, z jakichś ogólnych nakazów, lecz z nadgorliwości kierownictwa MSW. Uchwała RM z 1975 r. powoływała Rządowe Centrum Informatyczne PESEL jako jednostkę, która ma zaproponować sposób funkcjonowania systemu ewidencji ludności. Nie było tu żadnych tajemnic, lecz powolne, żmudne wklepywanie danych ewidencyjno-adresowych oraz gromadzenie danych z Magistra.

– Co zawiera obecnie i jak jest aktualizowana baza danych w PESEL-u?

– Baza zawiera dane ewidencyjne, a więc nazwisko i imię, stan cywilny, datę urodzenia, adres zamieszkania. Zbiorem objęta jest cała populacja i osoby zmarłe od 1980 r. Codziennie wprowadzane jest ok. 15 tys. aktualizacji. Większość to przemeldowania, 1200–1300 to rejestracja noworodków, 1000–1100 zgonów. Baza jest aktualizowana dzięki przesyłaniu informacji z biur meldunkowych z ok. 2,5 tys. gmin. PESEL nie rejestruje wykształcenia, miejsca pracy i zawodu, aczkolwiek to ostatnie byłoby interesujące z punktu widzenia potrzeb gospodarki. Docierają np. do nas zapytania od inwestorów, którzy chcieliby w dziewiczym terenie zbudować fabrykę i potrzebują fachowców określonej specjalności. Takich informacji aktualnych nie posiadamy. Mamy tylko stare dane, sprzed 10 lat.

– Tego rodzaju informacje były zawarte w odpowiedziach na ankiety wypełniane obowiązkowo przez osoby kończące wyższe studia, czyli w systemie Magister, o którym Pan zresztą wspomniał. Co się z nim obecnie dzieje?

– Dane z Magistra trafiały do rejestru do 1989 r., później praktycznie od razu były odkładane do archiwum. Od 1991 r. faktycznie przestał on funkcjonować, chociaż niektórzy kadrowcy jeszcze przysyłali ankiety. Zakres informacji w systemie Magister był tak duży, że wymagałoby to co najmniej dziesięciokrotnie większych środków finansowych, osobowych i technicznych, by można je było aktualizować. Znajdowały się tam m.in. informacje o dodatkowych kwalifikacjach, poziomie zarobków, publikacjach, znajomości języków obcych itp. Trafiały one do bazy danych tylko raz i nigdy nie były aktualizowane.

– Z jakim wyposażeniem technicznym i technologicznym startował przed blisko dwudziestu laty PESEL i jakie jest ono obecnie?

– Zaczęło się od zakupu dwóch dużych maszyn Siemensa. W 1980 r. na jednej z nich uruchomiono terenowy Bank Danych Warszawa, który do tej pory praktycznie na niej działa. Na drugiej zaczęto robić Centralny Bank Danych, zawierający informacje ewidencyjno-adresowe. Kolejny etap to rozpoczęty w 1984 r. i zakończony w 1985 r. projekt systemu obsługi własnej bazy danych JANTAR. Wykonana jest od początku do końca przez naszych programistów w asemblerze na maszynie Siemensa. Jest to system w modelu hierarchicznym, rozwiązania technologiczne, softwarowe zostały dostosowane do typu danych, które planowano gromadzić. Drugie założenie projektowe było takie, by baza, zwierająca bez mała 40 mln rekordów, mogła funkcjonować na maszynie o skromnych zasobach; 2 MB pamięci i 0,5 Mips. I to działało, choć czasem kiepsko. Przed rokiem zmieniliśmy maszynę na mainfraime IBM 4381. Głównym powodem wyboru było to, że asembler był identczny, co pozwoliło nam przenieść oprogramowanie bazy danych z niewielkimi zmianami. Jest to model dwuprocesorowy z 50 GB zasobami pamięci dyskowej, ale też już nieco przestarzały. Planujemy usprawnienie maszyny, bo zwiększy się liczba użytkowników i prawdopodobnie będziemy przechowywać dane graficzne – zdjęcia i podpisy. Planujemy też przejście na relacyjną bazę danych. Wyboru jeszcze nie dokonaliśmy, napierw chcemy ustalić ścieżkę rozwoju i sposób przenoszenia danych. Interesuje nas maszyna wieloprocesorowa, która pozwala uzyskać szybką informację i przedstawić ją użytkownikowi.

– Premier Pawlak w swoim wystąpieniu w Łańsku we wrześniu br., gdzie przedstawiał założenia rządowej strategii w zakresie tworzenia systemu informacyjnego państwa, stwierdził, iż w PESEL-u będą rejestrowani również cudzoziemcy znajdujący się nawet czasowo na terenie Polski.

– Nic nie stoi na przeszkodzie, by cudzoziemiec, który przebywa w Polsce, był zarejestrowany w PESEL-u. Chodzi przede wszystkim o to, by był objęty numerem ewidencyjnym i mógł rozliczać swoje podatki w Polsce, tak jak wszyscy obywatele.

– Jak dane są przekazywane do PESEL-a i w jaki sposób docierają do nich użytkownicy?

– Część gminy wysyła dokumenty do urzędów wojewódzkich, gdzie są rejestrowane w komputerze, ponad 900 gmin ma u siebie systemy komputerowe do wprowadzania danych. Wówczas przekazuje je na dyskietce do województwa. Jeśli w województwie istnieje bank wojewódzki lub terenowy, to dane są rejestrowane i częściowo metodą teletransmisyjną, a częściowo na nośnikach magnetycznych przekazywane do centralnej bazy danych. Jeśli w województwie banku danych nie ma – to wtedy rejestrowane są bezpośrednio z terminali do Centralnego Banku Danych. Użytkownicy korzystają z sieci PESEL-a. Funkcjonuje ona już od wielu lat, w tej chwili na łączach dzierżawionych, o architekturze gwiaździstej. Była to pierwsza w kraju sieć tak rozległa. U użytkownika jest albo terminal alfanumeryczny, albo PC z odpowiednim emulatorem. Teraz chcemy zmodernizować sieć teletransmisji i przejść na sieć pakietową. Na głównych kierunkach w kraju chcemy doprowadzić do używania łączy o przepustowości

Mgr Dariusz Kupiecki ukończył w 1978 r. Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Pierwszą pracę zawodową podjął w Zakładzie Translatorów Instytutu Maszyn Matematycznych. W 1984 r. przeszedł do Instytutu Podstaw Informatyki PAN, gdzie brał udział w pracach nad budową polskiego Unixa. W 1991 r. został zaangażowany do Urzędu Rady Ministrów, od 1992 r. jest w Rządowym Centrum Informatyki PESEL.

64 kB, co nam pozwoli dołączyć więcej użytkowników i usprawnić łączność.

– Informatyzacja administracji państwowej, zwłaszcza w terenie, przebiegała do niedawna w sposób bardzo chaotyczny, co owocowało brakiem standardów, a ty samym ogromnie utrudniało porozumiewanie się między poszczególnymi jednostkami. Czy ten brak standardów nęka też PESEL?

– Nie. W naszym przypadku gmina z gminą, jak i z nami, od początku porozumiewa się bez trudu, bo rozmawia naszym językiem. Nasze standardy narzucane są przez technologię. Na początku stosowany był protokół siemensowski, klasy BSC, ale z pewnymi zmianami. Teraz mamy protokoły IBM-owskie. Określamy też standardy tworzenia informacji i gospodarowania nią, a więc model danych.

– PESEL mieści się w pilnie strzeżony budynku, czy równie dobrze strzeżony jest dostęp do danych osobowych zarejestrowanych w PESEL-u? Kim są jego użytkownicy?

– Użytkowników dzielimy na dwie grupy: administracja państwowa i inni. Jeśli chodzi o administrację, to jednostka, która się do nas zgłasza jest sprawdzana czy w przepisach regulujących jej działanie jest zapisana konieczność gospodarowania danymi osobowymi. Jeśli tak, to wówczas wydajemy zgodę i taki urząd dostaje terminal lub otrzymuje potrzebną mu informację. Jeśli dostaje terminal, to w haśle określamy, do jakich danych może mieć dostęp. Nikt poza nami nie ma możliwości i prawa aktualizować danych. Dostęp ma policja, UOP, Urzędy Wojewódzkie, Biura Paszportowe, wojskowe komendy uzupełnień (uczestniczymy w przygotowywaniu list poborowych i przedpoborowych), GUC, Straż Granicza. Natomiast innym przekazujemy dane zbiorcze. Np. kuratoriom oświaty i szkołom przygotowujemy dane w zakresie dotyczącym siedmiolatków, do GUS przekazujemy informacje statystyczne, udostępniamy określone dane potrzebne do badań naukowych określonych grup ludności, np. na temat zdrowia, dane do badań opinii publicznej, a więc otrzymuje je OBOS, CBOS, Demoskop itp. Natomiast odmawialiśmy, odmawiamy i będziemy odmawiać odpowiedzi na pytania tego rodzaju z firm, które zajmują się reklamą bezpośrednią.

– A jednak trafia do domów natrętna reklama bezpośrednia, a tygodnik „Milioner" drukuje numery dowodów, również osób nieżyjących. Takich danych nie czerpie się z książek telefonicznych...

– Można do nich dotrzeć z pominięciem PESEL-a. Duże zbiory danych mają m.in. Zakłady Energetyczne, Telekomunikacja Polska S.A., rejestr pojazdów. Wiemy, że niektóre kliniki położnicze udostępniają dane firmom działającym na rynku artykułów dla dzieci. „Miliarderowi" podaliśmy jedynie przedziały numerów dowodów, natomiast nie udostępniliśmy redakcji informacji o tym, które numery już nie funkcjonują.

– Mamy PESEL, REGON, tworzy się TERYT, czyli informacja o terenie i mamy mieć Infostradę rządową, czyli sieć ogólnokrajową, w której będą krążyć gigabajty danych, również osobowych. Czy nie sądzi Pan, że zbliżamy się do świata kreowanego przez Orwella w „1984"? Łączenie danych z różnych zbiorów, co umożliwia informatyzacja, to duże zagrożenie dla wolności jednostki.

– Obecnie jesteśmy bliżej tego świata niż, jak sądzę, możemy być w przyszłości. W tej chwili w polskim prawodawstwie, nie ma bowiem żadnej regulacji prawnej dotyczącej ochrony danych osobowych. PESEL chroni je na zasadzie dobrej woli. U innych można je dziś bez sankcji karnych otrzymać, kupić. Zdając sobie sprawę z zagrożenia stworzyliśmy projekt ustawy o ochronie danych osobowych, który – mamy nadzieję – wkrótce trafi do Sejmu. Jest on wzorowany na istniejącym ustawodawstwie krajów Wspólnoty Europejskiej i został już przedstawiony na konferencji w Strasburgu. Jeśli ustawa zostanie ratyfikowana przez Sejm, Polska może podpisać konwencję 108 Rady Europy, która przedstawia zalecenia dla państw członkowskich, jak należy regulować tę dziedzinę. Wówczas też będzie można zakazać np. zakładowi energetycznemu udostępniania danych, ograniczyć dostęp do rejestru pojazdów czy do spisu rozmów telefonicznych. Chcielibyśmy bardzo wysoko umieścić organ, który by nadzorował wykonywanie tej ustawy, wręcz stworzyć coś w rodzaju rzecznika danych osobowych, wybieranego przez Sejm, mającego prawo kontrolować co dzieje się z danymi osobowymi w administracji rządowej. W jego urzędzie powinien być prowadzony rejestr zbiorów.

– Podjęte zostały prace nad statusem dokumentu elektronicznego. Czy PESEL jest też tym zainteresowany?

– Pracuję obecnie nad określeniem statusu komunikatów EDI dotyczących zdarzeń z ewidencji ludności. Komunikaty te umożliwiłyby przepływ danych między innymi zbiorami a PESEL-em. Chodzi o duże zbiory prowadzone w administracji, m.in. ZUS, POLTAX, urzędy pracy. Korzystałyby z nich także banki, które powinny mieć prawo sprawdzenia czy kredytobiorca to rzeczywiście ta osoba, która się legitymuje danym dowodem. Taki jest obraz sytuacji, która się utworzy w przyszłości. PESEL na pewno będzie zawsze systemem służebnym dla wszystkich pozostałych. Nie czerpie bowiem informacji z żadnych innych zbiorów osobowych, natomiast jest takim źródłem informacji dla innych.

– Dziękuję Panu za rozmowę.

Stan świadomości

To była udana impreza – stwierdzali zgodnie uczestnicy i goście I Kongresu Informatyki Polskiej, który odbył się w Poznaniu w dniach 1–3 grudnia br.

Zasługa to Komitetu Programowo-Organizacyjnego, utworzonego przez cztery organizacje działające w środowisku: Polską Izbę Informatyki i Telekomunikacji, Polskie Towarzystwo Informatyczne, Stowarzyszenie Rozwoju Systemów Otwartych i Stowarzyszenie Polski Rynek Oprogramowania „PRO". W okresie poprzedzającym Kongres zorganizowały one wiele imprez, na których dyskutowano o problemach nurtujących poszczególne środowiska informatyczne. Przygotowano też analizy aktualnego stanu sytuacji i świadomości środowiska. Sesje robocze toczyły się w pięciu grupach tematycznych (Rządowe przedsięwzięcia tematyczne, Warunki rozwoju polskiego oprogramowania, Świadomość społeczna w odbiorze informatyki, Wypełnienie potrzeb kadrowych w informatyce, Nowe technologie teleinformatyczne) oraz w trzech grupach środowiskowych (Użytkownicy, Producenci i Dostawcy, Edukacja i Szkolenie). Celem Kongresu było znalezienie odpowiedzi na pytanie, jakie interesy łączą, a jakie dzielą te środowiska. Zgodnie z apelem przewodniczącego Komitetu Organizacyjnego, dr. inż. Wacława Iszkowskiego, uczestnicy Kongresu nie narzekali, lecz rzeczowo omawiali szanse i zagrożenia informatyki w Polsce.

Mówiono o sprawach, które są wszystkim znane, ale po raz pierwszy, jak wielokrotnie podkreślano, udało się je uporządkować i dokładnie opisać.

Dla użytkowników celem zastosowania informatyki jest przede wszystkim zwiększenie efektywności działania instytucji, wspomaganie decyzji przez dostęp do informacji o przyspieszonym obiegu. Najbardziej pozytywnym czynnikiem jest dostęp do najnowszych technologii, zaś największym zagrożeniem brak polskich firm integrujących systemy komputerowe. Dla środowiska zajmującego się nauką i edukacją ważny jest dostęp do najnowszych technologii i światowych zasobów informacji, zagrożeniem zaś pauperyzacja tego środowiska. Może ona zaowocować w niedalekiej przyszłości gorszym przygotowaniem kadr dla informatyki, a nawet jej brakiem. Z tego zagrożenia zdają sobie sprawę zarówno użytkownicy jak i trzecia grupa uczestników Kongresu: producenci i dostawcy. Celem ich jest wprowadzanie nowych technologii i zwiększanie zysków. Atutem są rosnące potrzeby użytkowników, zagrożeniem zaś jest niezbyt precyzyjne sformułowanie potrzeb informatycznych użytkowników, niestabilna finansowo gospodarka, zbyt duże obciążenia podatkowe i celne.

W ostatnim dniu Kongresu z jego uczestnikami spotkał się premier Waldemar Pawlak.

Wnioski Kongresowe oraz opinie jego organizatorów i uczestników szerzej przedstawimy w styczniowym numerze „Informatyki". Z ogromną satysfakcją odnotowujemy, iż prezes PTI, Piotr Fuglewicz, na sesji roboczej grupy środowiskowej „Edukacja i szkolenia" zapraszał jej uczestników do współpracy z naszą redakcją.

(kar)

POSTGRES jako narzędzie budowy aktywnych i dedukcyjnych systemów baz danych

MARCIN JANUCHTA
ZBYSZKO KRÓLIKOWSKI
Instytut Informatyki
Politechnika Poznańska

Większość współczesnych systemów baz danych to systemy pasywne, w których wszystkie operacje manipulowania danymi są związane tylko i wyłącznie z realizacją żądań użytkowników. Jeśli natomiast system bazy danych jest wyposażony w mechanizmy definiowania i przetwarzania reguł produkcyjnych [1] (ang. *production rules*, w tym kontekście również – *triggers* lub *alerters*), to jest to system aktywnej bazy danych [3, 4, 6, 8]. Reguły te są uaktywniane w wyniku wykonania niektórych operacji na bazie danych, wystąpienia określonego stanu bazy danych lub relacji między tymi stanami, jak również niektórych innych zdarzeń. Reguły produkcyjne stanowią jednorodny i spójny mechanizm, umożliwiający z poziomu systemu zarządzania bazą danych egzekwowanie więzów integralności i praw dostępu, monitorowanie dostępu do danych, śledzenie ewolucji bazy danych oraz otrzymywanie danych pochodnych. Dodatkowo, mechanizm wnioskowania z użyciem reguł produkcyjnych stwarza możliwości budowy dużych i efektywnych systemów baz wiedzy.

W dotychczasowych systemach, jeśli takie możliwości istniały, to były związane z aplikacjami użytkowników, a co więcej, dla każdej z nich stosowano odrębne, nie powiązane między sobą rozwiązania. Odpowiednio przygotowane programy aplikacyjne mogły być uruchamiane z określoną częstotliwością w celu sprawdzenia, czy określone warunki są spełnione w systemie. Jest to mechanizm tzw. odpytywania bazy danych (ang. *polling*). Jednakże, jeśli częstotliwość uruchamiania takich programów będzie zbyt duża, to może spowodować zmniejszenie efektywności systemu. Jeśli z kolei częstotliwość ta będzie zbyt mała, to niektóre sytuacje mogą zostać nie wykryte. Inne rozwiązanie, tzn. włączenie modułów sprawdzających określone warunki do wszystkich programów dokonujących modyfikacji bazy danych, jest nie do przyjęcia z punktu widzenia inżynierii oprogramowania, ponieważ jego złożoność rośnie w sposób nieakceptowalny.

Rozwiązaniem przyszłościowym, pozbawionym wyżej wspomnianych wad, jest integracja reguł z systemem zarządzania bazą danych. W chwili obecnej niektóre elementy mechanizmu reguł produkcyjnych zostały włączone do produktów komercyjnych, takich jak: INGRES, ORACLE, Rdb i SYBASE. Jednakże, proponowane w systemach eksperymentalnych rozwiązania są znacznie bardziej zaawansowane [4–7, 8, 14, 17, 21]. W ostatnich latach są prowadzone intensywne prace w ramach kilku dużych projektów badawczych, których celem jest m.in. budowa efektywnych systemów aktywnych baz danych. Jednym z wiodących projektów z tej dziedziny jest system POSTGRES [9–18].

Praca częściowo finansowana w ramach grantu 43-0329 (KBN2).

Budowa systemu aktywnej bazy danych z wykorzystaniem systemu POSTGRES

Reguły produkcyjne mają określoną składnię oraz mechanizm wnioskowania. Tworzą one łącznie tzw. **język regułowy**. Języki reguł produkcyjnych, implementowane w różnych systemach aktywnych baz danych, różnią się znacznie z punktu widzenia ich semantyki i syntaktyki oraz wykorzystywania mechanizmów wnioskowania [6]. Korzenie języków regułowych baz danych wywodzą się ze sztucznej inteligencji [2]. Reguły definiowane w ramach tej dziedziny wiedzy przyjmują następującą postać:

wzorzec → *akcja*

Reguła jest uaktywniana wtedy, gdy **wzorzec** (ang. *pattern*) „pasuje" do danych znajdujących się w przestrzeni pamięci roboczej systemu, natomiast **akcja** (ang. *action*) dokonuje modyfikacji pamięci roboczej, zgodnie z dopasowaniem danych. Wzorzec jest również określany mianem warunku (ang. *condition*) lub predykatu (ang. *predicate*). Innymi słowy, przyjmuje się założenie, że reguła jest uaktywniana tylko wtedy, gdy w pamięci roboczej pojawią się nowe dane pasujące do wzorca (lub w przypadku reguł zanegowanych, gdy pasujące dane są usuwane z pamięci roboczej). Tak więc reguły pośrednio są uaktywniane przez zdarzenia, takie jak wprowadzenie nowych danych oraz modyfikacje lub usunięcie danych.

Język regułowy aktywnych baz danych

Podstawowa różnica pomiędzy językami regułowymi sztucznej inteligencji a językami stosowanymi w systemach baz danych polega na tym, że **zdarzenia** (ang. *events*) uaktywniając reguły są specyfikowane w sposób bezpośredni [1, 3]. Reguły w systemach baz danych przyjmują następującą postać:

on *zdarzenie*
if *warunek*
then *akcja*

Reguła jest uaktywniana wtedy, gdy wystąpi określone zdarzenie. Warunek zdefiniowany w treści reguły jest sprawdzany na podstawie danych. Jeśli warunek jest spełniony, podejmowana jest określona akcja.

Powyższy sposób reprezentacji reguł, zaproponowany w [8], stał się standardem znanym jako reguły ECA (ang.

Event-Condition-Action). Szczegóły i stopień złożoności specyfikacji zdarzeń, warunków i akcji różnią się jednak znacznie w poszczególnych systemach. W niektórych językach dane związane ze zdarzeniem i/lub częścią warunkową mogą być przekazywane do części warunkowej i/lub do definicji akcji. Niektóre języki (np. w systemie Starburst [21]) są wyposażone w mechanizmy porządkowania reguł (ang. *rule ordering*), których celem jest określenie, jaka reguła powinna być wykonana w sytuacji, gdy wiele reguł zostało uaktywnionych (ang. *conflict resolution*).

Specyfikacja zdarzeń

Praktycznie wszystkie systemy aktywnych baz danych stwarzają możliwość definiowania reguł uaktywnianych pośrednio lub bezpośrednio przez modyfikacje bazy danych [5]. Większość zdarzeń auktywniających reguły stanowią właśnie modyfikacje. W systemach relacyjnych baz danych są to następujące operacje: **insert, delete, update**, natomiast w systemach obiektowych baz danych modyfikacje są związane z wywołaniem metody na monitorowanym obiekcie.

W systemach relacyjnych i postrelacyjnych reguły uaktywniane przez modyfikacje bazy danych mogą mieć następującą postać:

```
define rule Kontrola_Zatrudnienia is
on append to Pracownik
where ...
then ...
```

gdzie *Pracownik* jest relacją (zgodnie z terminologią systemu POSTGRES – klasą) zawierającą informacje o pracownikach.

Większość języków regułowych stwarza możliwość uaktywniania reguł na podstawie operacji dostępu do danych. Jeszcze inny typ zdarzeń uaktywniających, to zdarzenia czasowe (ang. *timing events*), co oznacza, że reguły mogą być uaktywniane w określonym czasie lub przedziale czasu. Powyższe zdarzenia zaliczamy do zdarzeń prostych. Zdarzenia proste mogą pochodzić również z innych źródeł: z systemu operacyjnego, przemysłowej instalacji lub sieci komputerowej (np. zdarzenie „nowy użytkownik").

Niektóre języki [4, 5, 7] stwarzają możliwość definiowania zdarzeń złożonych (ang. *composite events*), którymi mogą być kombinacje zdarzeń czasowych i dostępu do danych np. ich sekwencja, alternatywa lub wystąpienie określonego zbioru zdarzeń, jak również tzw. zdarzenia okresowe.

W eksperymentalnym SZBD POSTGRES, zdarzeniami uaktywniającymi reguły mogą być operacje wprowadzania, usuwania, aktualizacji oraz wyszukiwania danych **(append, delete, replace, retrieve)**. Warto dodać, że w innych systemach są używane inne określenia, np. **insert** zamiast **append** oraz **update** zamiast **replace**.

Specyfikacja warunków

We wszystkich językach regułowych aktywnych baz danych, część warunkowa określa warunek lub zapytanie dotyczące informacji przechowywanych w bazie danych. Warunek może być pominięty w definicji reguły. W wielu językach regułowych warunki mogą odnosić się do modyfikowanych danych lub stanu bazy danych poprzedzającego modyfikacje. Istnieje wówczas możliwość definiowania **warunków przechodnich** (ang. *transition conditions*), tzn. warunków definiowanych na podstawie zmian stanu bazy danych.

W systemie POSTGRES warunki są definiowane na podstawie aktualnego stanu bazy danych lub na podstawie zmian między stanami [10–15, 17]. Np. reguła:

```
define rule Monitorowanie_Pracownikow is
on replace to Pracownik
where Pracownik.nazwisko = "M.Januchta"
then ...
```

bada aktualny stan bazy danych.
Natomiast reguła:

```
define rule Kontrola_Wzrostu_Plac is
on replace to Pracownik
where new.pensja > 1.1 * current.pensja
then ...
```

wykorzystuje obie wspomniane możliwości. Reguła ta jest uaktywniana wtedy, gdy uaktualniona (w wyniku operacji *replace*) wartość atrybutu *pensja* klasy *Pracownik* przekroczy o 10% wartość bieżącą *(current)*.

Specyfikacja akcji

Akcja reguły produkcyjnej określa operacje, jakie mają być wykonane, jeśli część warunkowa reguły jest spełniona. Języki regułowe systemów aktywnych baz danych pozwalają na definiowanie akcji o charakterze znacznie bardziej ogólnym niż w przypadku systemów eksperckich (por. [2]). Akcje mogą być sekwencjami operacji wyszukiwania i modyfikacji danych, mogą określać sposób wycofania bieżącej transakcji (ang. *rollback*) oraz mogą być wywołaniami procedur aplikacji. Akcja może odnosić się do danych, których modyfikacja spowodowała, że reguła została uaktywniona. Np. reguła:

```
define rule Kontrola_Zatrudnienia is
on append to Pracownik
do delete Pracownik where Pracownik.nazwisko = new.nazwisko
```

jest uaktywniana każdorazowo, gdy do bazy danych są wprowadzane informacje o nowych pracownikach. Akcja tej reguły polega na usunięciu „istniejącego" pracownika, jeśli nazwisko nowo wprowadzanego jest takie samo jak nazwisko pracownika, którego opis już się znajdował w bazie danych.

Związki między zdarzeniami, warunkami i akcjami

Związki między poszczególnymi elementami reguł występujące w językach regułowych systemów aktywnych baz danych są bardzo zróżnicowane. W większości systemów reguły są uaktywniane przez operacje wprowadzania, usuwania i aktualizacji danych zawartych w określonej relacji. Do każdej wprowadzanej, usuwanej lub aktualizowanej krotki może nastąpić odwołanie w części warunkowej i akcji reguły. Odwołanie w części warunkowej i akcji reguły może dotyczyć również całej relacji, w której nastąpiły zmiany. Wykorzystywany jest wówczas mechanizm zmiennych krotkowych, które przybierają wartości kolejnych obiektów z danej klasy.

W systemie POSTGRES w części warunkowej i w przypadku akcji, odniesienie przez nazwę klasy dotyczy klasy w całości. Innymi słowy, krotki przetwarzane w poszczególnych częściach reguły nie są ze sobą domyślnie powiązane. Np. reguła:

```
define rule Bez_podwyżki is
on replace to Pracownik.pensja
```

```
where new.nazwisko= "M.Januchta"
do replace new (pensja = 2000000)
```

powoduje zmianę pensji wszystkich pracowników. Aby zmienić pensję tylko pracownika „aktualizowanego", należy zdefiniować następującą regułę:

```
define rule Bez_podwyżki is
on replace to Pracownik.pensja
where Pracownik.nazwisko= "M.Januchta"
do replace Pracownik (pensja = 2000000)
```

Przy odwołaniu do zmiennej krotki wykorzystuje się zmienne krotkowe *new* i *current*. Dotyczą one tego wystąpienia w aktualizowanej klasie, którego aktualizacja uaktywniła regułę. Jeśli reguła jest uaktywniana przez operację **append**, to *new* dotyczy wartości wprowadzanej, a *current* pozostaje niezdefiniowane. Jeśli reguła jest uaktywniana przez operację **delete**, to *current* dotyczy wartości usuwanej, a *new* jest niezdefiniowane. Jeśli natomiast uaktywnienie następuje w wyniku operacji **replace**, wtedy *new* i *current* określają nową i starą wartość definiowanego wystąpienia. Na przykład reguła *Podział_Zarobków*:

```
define rule Podział_Zarobkow is
on delete to Pracownik
do replace Pracownik (pensja = pensja + 0.1 * current. pensja)
```

jest uaktywniana wtedy, gdy z klasy *Pracownik* jest usuwany opis jakiegoś pracownika. Akcja reguły polega na podniesieniu zarobków pozostałych pracowników o 10% pensji pracownika usuwanego.

Definiowanie reguł w systemie POSTGRES

Składnia polecenia wykorzystywanego do definiowania reguł w systemie POSTGRES przedstawia się następująco:

```
define [ instance | rewrite ] rule nazwa_reguły
    [ as exception to nazwa_reguły_n ] is
    on zdarzenie
    to obiekt [ [ from klauzula ] where klauzula ]
    do [ instead ] [ akcja | nothing [ akcja ] ]
```

Zdarzeniem może być jedno z poleceń: **retrieve, replace, delete** lub **append**. Obiektem może być klasa lub jej atrybut. Klauzule występujące po słowach kluczowych **from** i **where** są zdefiniowanymi wcześniej klauzulami języka zapytań POSTQUEL. Akcja może być jednym z następujących poleceń języka POSTQUEL: **retrieve, replace, delete** lub **append**. W części warunkowej reguły, wystąpienia, na których mają być wykonane operacje aktualizacji, mogą być oznaczone *current* – „bieżąca, stara wersja krotki" lub *new* – „nowa wersja".

Akcja może spowodować wykonanie takiego samego polecenia, które uaktywniło regułę. W związku z tym, akcja reguły może być wyróżniona znacznikiem *instead*. Bez tego znacznika akcja będzie wykonywana wraz z poleceniem uaktywniającym regułę. Jeszcze inną możliwość stwarza słowo kluczowe *nothing*. Powoduje ono, że nie będą podejmowane żadne akcje. Użycie *instead nothing* powoduje, że również nie zostanie wykonana operacja, która spowodowała uaktywnienie reguły.

Rozważmy kilka przykładowych reguł zdefiniowanych dla naszej przykładowej bazy danych.

```
define rule Regulacja_Płac is
on replace to Pracownik.pensja
where current.nazwisko = "M.Januchta"
do replace Pracownik (pensja = new.pensja) where
    Pracownik.nazwisko = "M.Matysiak"
```

Zdarzeniem w tej regule jest operacja aktualizacji. Część warunkowa definiuje atrybut klasy *Pracownik* oraz konkretne wystąpienie tej klasy, których aktualizacja dotyczy. Akcja tej reguły powoduje wstawienie nowych wartości pensji pracowników: M. Januchta i M. Matysiak. Innymi słowy, ilekroć będzie zmieniana pensja pracownika M. Januchta, to w konsekwencji będzie to powodować zmianę pensji M. Matysiaka.

```
define rule Płace_Nowych_Pracowników_1 is
on append to Pracownik where new.pensja > 3 000 000
do replace new (pensja = 3 000 000)
```

W tym przypadku, zdarzeniem jest operacja wprowadzania do klasy *Pracownik* nowego wystąpienia. Warunek określa, że jeśli wprowadzona pensja nowego pracownika będzie wyższa niż 3 000 000, to powinna być podjęta zdefiniowana akcja. Akcja powoduje wstawienie wartości pensji równej 3 000 000 w miejsce wartości wstawionej uprzednio.

```
define rule Utajnienie_Zarobków is
on retrieve to Pracownik.pensja where Pracownik.nazwisko =
    "Z.Krolikowski" and pg_username() = "M.Januchta"
do instead nothing
```

W tym przypadku zdarzeniem jest operacja wyszukiwania danych. Część warunkowa definiuje atrybut klasy *Pracownik* (tutaj pensja) oraz konkretne wystąpienie tej klasy – pracownik Z. Królikowski, których operacja wyszukiwania dotyczy. Dodatkowo w części warunkowej zdefiniowano użytkownika, który te operacje wykonuje – funkcja *pg_username* ( ) zwraca wartość identyfikatora użytkownika wykonującego daną operację *retrieve*. Akcja tej reguły określa, że jeśli jest podejmowana próba wykonania operacji zdefiniowanej w części warunkowej reguły, przy założeniu, że operację tę chciałby wykonać M. Januchta, to system nie powinien wykonać żadnych czynności. Innymi słowy, reguła powyższa chroni informacje o zarobkach określonych pracowników przed niepowołanym dostępem ze strony innych użytkowników systemu. Niestety, aktualna wersja systemu POSTGRES nie pozwala na wywoływanie funkcji w regułach.

Zarządzanie regułami

Jednym z najistotniejszych problemów zarządzania regułami jest kwestia wyboru reguły do wykonania w sytuacji, gdy w wyniku wystąpienia określonego zdarzenia zostanie uaktywnionych wiele reguł [1, 6]. W niektórych systemach wybór reguły jest wykonywany arbitralnie przez system, w innych użytkownik definiujący reguły może wpływać na rozwiązywanie konfliktów między regułami. Reguły mogą być częściowo uporządkowane, tzn. dla dwóch reguł, jedna z nich jest oznaczana jako posiadająca wyższy priorytet. W systemie POSTGRES badane są możliwości wykorzystania priorytetów numerycznych oraz hierarchii wyjątków (fraza: **as exception to nazwa_reguły_n** w definicji reguły) dla rozwiązania problemu wyboru reguł [14, 15, 17]; jednakże w wersji 4.1 tego systemu, żadne z tych rozwiązań nie zostało jeszcze zaimplementowane.

Semantyka przetwarzania reguł określa interakcję reguł z operacjami na bazie danych, transakcjami użytkowników oraz programami aplikacyjnymi. Nawet dla małego zbioru reguł ich zachowanie może być bardzo złożone i nieprzewidywalne. Istnieje wiele sposobów wykonywania reguł. W systemach

eksperckich wykorzystywany jest najczęściej algorytm: *recognize – act cycle* (z systemu OPS5 [2]).

W systemach aktywnych baz danych, gdzie przetwarzanie reguł jest w pełni zintegrowane z konwencjonalnymi działaniami na bazie danych, semantyka *recognize – act*, implementowana w czystej postaci, nie jest niestety odpowiednia. Co więcej, w tych systemach algorytm przetwarzania reguł powinien uwzględniać przyczynę uaktywnienia systemu regułowego. W systemie relacyjnej bazy danych operacje mogą być wykonywane na zbiorach, np. do relacji jest wprowadzany zbiór krotek spełniających określone warunki. Operacje wielokrotne są grupowane w postaci transakcji. Tak więc, przetwarzanie reguł może być wywołane przez zmiany dotyczące pojedynczej krotki (ang. *single tuple_level changes*) lub przez transakcje. Jest to problem tak zwanej **ziarnistości reguł** (ang. *granularity*).

W systemie POSTGRES, inaczej niż w innych systemach aktywnych baz danych, system regułowy jest uruchamiany natychmiast po wykonaniu próby dostępu do krotki, jeśli operacja ta spełnia warunki jednej lub więcej reguł. Mówimy o przetwarzaniu zorientowanym na krotkę. W niektórych innych systemach [5, 6] przetwarzanie reguł ma odmienny charakter – jest zorientowane na zbiór krotek (ang. *set-oriented*). Gdy akcja reguły jest wykonywana, może ona modyfikować wiele dodatkowych krotek, co z kolei natychmiast uaktywnia ponownie te same reguły lub reguły dodatkowe. W systemie POSTGRES przetwarzanie reguł jest rekursywne, w przeciwieństwie do przetwarzania sekwencyjnego wykorzystywanego w innych systemach.

Jako ilustrację różnic między przetwarzaniem zorientowanym na krotki i zbiory krotek, rozważmy następujący przykład:

```
define rule Płace_Nowych_Pracowników_2 is
on append to Pracownik
do replace new (placa = 0.9 * int4ave(Pracownik.placa))
```

gdzie **int4ave** oznacza obliczenie średniej z argumentów typu *int4*. Reguła ta jest uaktywniana przy wykonywaniu operacji wprowadzania nowych wystąpień do klasy *Pracownik*. Załóżmy, że do tej klasy jest wprowadzana grupa nowych pracowników. Przy przetwarzaniu zorientowanym na krotki (jak w systemie POSTGRES), reguła powyższa jest uaktywniana dla każdego nowo wprowadzanego pracownika, a więc pensje nowych pracowników z tej grupy będą się różnić. Przy przetwarzaniu zorientowanym na zbiory krotek, reguła byłaby uaktywniona tylko raz – pensje nowych pracowników byłyby takie same.

Podstawowy algorytm przetwarzania reguł w systemie POSTGRES przedstawia się następująco:

1. W efekcie działań użytkownika lub wykonania reguły następuje żądanie dostępu do wystąpienia (do krotki) w danej klasie.
2. Zgłoszenie żądania uaktywnia reguły i jeśli warunki reguł $R_1, ..., R_n$ są spełnione, to
3. Dla każdej uaktywnionej reguły R_i jest wykonywana jej akcja.

Akcja może spowodować modyfikację wystąpienia w określonej klasie i rekursywne wywołanie powyższego algorytmu. Jeśli jako akcja reguły zostałoby zdefiniowane wycofanie transakcji, to przetwarzanie reguł jest zatrzymywane, a transakcja jest wycofywana.

W systemie POSTGRES zastosowano dwa mechanizmy przetwarzania reguł [10, 18]: przedstawiony wyżej schemat przetwarzania zorientowany na krotki (deklarowany za pomocą słowa kluczowego *instance* w poleceniu *define rule*) oraz mechanizm modyfikacji zapytań (ang. *query rewrite*), deklarowany za pomocą słowa kluczowego *rewrite* w poleceniu *define rule*. Tak więc, podczas definiowania reguły użytkownik może wybrać, który mechanizm przetwarzania należy zastosować dla danej reguły.

Implementacja mechanizmu zorientowanego na krotki przedstawia się następująco. Przy każdym wystąpieniu danej klasy (w ujęciu tradycyjnym – dla każdej krotki relacji), dla którego spełnione są warunki zdefiniowane w regule, jest wstawiany odpowiedni znacznik. Ponieważ reguł, dla których są spełnione warunki, może być kilka, z każdym takim wystąpieniem może być związanych kilka znaczników. Jeśli wystąpienie danej klasy jest modyfikowane lub wyszukiwane, i jest oznaczone jednym lub kilkoma znacznikami, wówczas odpowiednia reguła lub reguły są lokalizowane (na podstawie znaczników) i ich akcje wykonywane. Znaczniki muszą być instalowane i usuwane wtedy, gdy reguły i dane są tworzone, usuwane i modyfikowane. Wprowadza to niestety dodatkowy narzut czasowy na wykonywanie tych operacji. Stosowanie tego mechanizmu zaleca się wtedy, gdy został zdefiniowany zbiór reguł dotyczących danej klasy, a każda z tych reguł dotyczy małego zbioru wystąpień.

Mechanizm *query rewrite* jest efektywniejszy, jeśli reguła dotyczy dużego zbioru danych. Mechanizm ten został zaimplementowany przez wbudowanie między parserem a procesorem zapytań dodatkowego modułu, który „przechwytuje" polecenia użytkowników i zmienia je (stąd nazwa tego mechanizmu – ang. *rewrite*) zwiększając o dodatkowe polecenia, które stanowią akcję reguły uaktywnionej przez oryginalne polecenie. Zwróćmy uwagę, że to dodatkowe polecenie może również uaktywnić regułę. Mechanizm *query rewrite* jest stosowany podczas kompilacji zapytań, natomiast mechanizm zorientowany na krotki, podczas interpretacji zapytań *ad hoc*.

Budowa systemu dedukcyjnej bazy danych z wykorzystaniem systemu POSTGRES

Koncepcję oraz aktualny stan rozwoju systemów dedukcyjnych baz danych przedstawiono już na łamach INFORMATYKI [22]. Tutaj skoncentrujemy się na analizie możliwości budowy takich systemów z wykorzystaniem SZBD POSTGRES.

System dedukcyjnej bazy danych zawiera oprócz zwykłych relacji reguły, które pozwalają na zadawanie skomplikowanych zapytań, trudnych do wyrażenia przez standardowe języki typu SQL. Dedukcyjna baza danych zawiera dwa typy relacji: relacje rzeczywiste (zwykłe relacje, których zawartość jest przechowywana w bazie danych) oraz relacje wirtualne, które są konstruowane na podstawie podanych reguł. Reguły określają powiązania między tymi relacjami.

W dedukcyjnych bazach danych często używanym językiem reguł jest Datalog [20]. Język ten jest związany z relacyjnym modelem danych. W wyrażeniach Datalogu nie używa się nazw atrybutów relacji; zamiast tego są one identyfikowane na podstawie ich kolejności w nagłówku wywołania. Nazwy reguł odpowiadają nazwom relacji. Algorytm przekształcania i opis języka Datalog znajduje się w [20]. Opis implementacji translatora Datalogu na język SQL SZBD ORACLE 7, zamierzamy przedstawić na łamach INFORMATYKI w najbliższym czasie.

Poniżej przedstawimy przykład wykorzystania dedukcji w systemie POSTGRES. Rozważmy następującą, klasyczną już regułę języka Datalog, obliczającą relację wirtualną

Dziadkowie na podstawie relacji *Rodzice:*
Dziadkowie (X, Y): – *Rodzice* (X, Z), *Rodzice* (Z, Y).

Obliczenie wyniku polega na podłączeniu dwu kopii (ang. *self-join*) relacji *Rodzice* i wybraniu odpowiednich atrybutów. Aby zrealizować powyższą operację w systemie POSTGRES, należy wydać następujące zapytanie:

```
retrieve (p1.dziecko, p2.rodzic)
from p1, p2 in Rodzice
where p1.rodzic = p2.dziecko
```

które odpowiada następującej kombinacji operacji projekcji, selekcji i połączenia, wynikającej z przekształceń wyrażenia języka Datalog do postaci algebry relacji:

$$\Pi_{p1.\$1,p2.\$2}(\sigma_{p1.\$2=p2.\$1}(p1 \rhd\lhd p2))$$

Aby można było korzystać bezpośrednio z wirtualnej, nie istniejącej na stałe, relacji *Dziadkowie*, należy wszystkie zapytania do tej relacji zastąpić wykonaniem powyższego zapytania. Można to osiągnąć definiując następującą regułę:

```
define rule dziadkowie is
on retrieve to Dziadkowie
do instead  retrieve (p1.dziecko, p2.rodzic)
            from p1, p2 in Rodzice
            where p1.rodzic = p2.dziecko
```

Teraz każdy odczyt z relacji *Dziadkowie* będzie zastąpiony przez wykonanie powyższej reguły, a konkretnie, zapytania zdefiniowanego po słowie kluczowym *instead.*

W powyższym rozwiązaniu nie mamy możliwości dopisywania krotek bezpośrednio do relacji *Dziadkowie*. Jeżeli chcemy uzyskać relację mieszaną, tj. relację, w której część danych jest reprezentowana eksplicite w postaci krotek, zaś reszta jest obliczana na podstawie reguły, to w definicji reguły możemy zrezygnować ze słowa kluczowego *instead.* W ten sposób uwzględniamy zarówno wynik działania reguły, jak i bezpośrednio wpisane dane. Taką definicję dla reguły Datalogu *Rodzeństwo* (X, Y): – *Rodzice* (Z, X), *Rodzice* (Z, Y).

stanowi następująca reguła systemu POSTGRES:

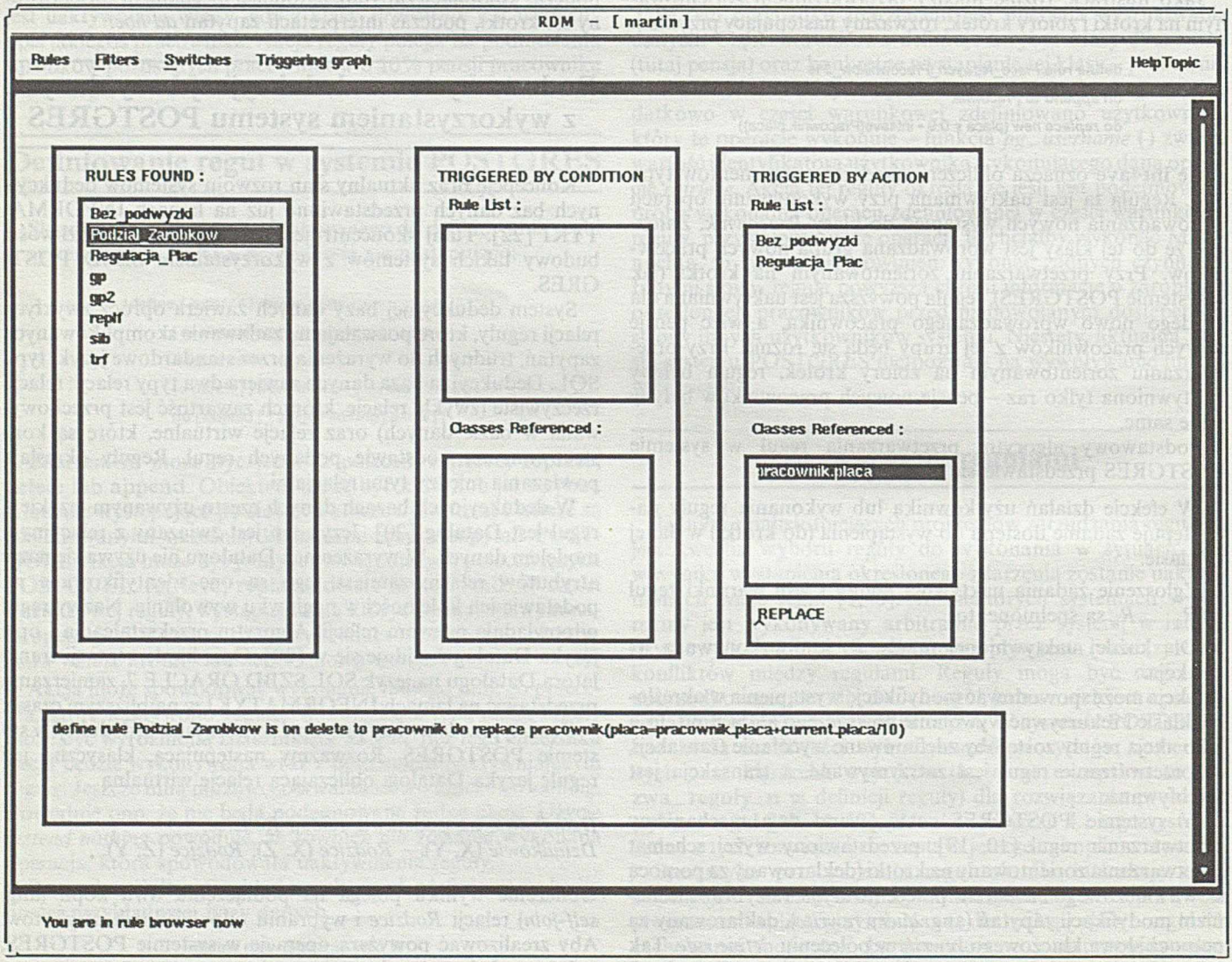
```
define rule rodzenstwo is
on retrieve to Rodzenstwo
do    retrieve (rodzenstwo_1 = p1.dziecko, rodzenstwo_2 = p2.dziecko)
      from p1, p2 in Rodzice  where p1.rodzic = p2.rodzic
```

Oprogramowanie wspomagające system regułowy

Definiowanie reguł w systemie POSTGRES jest w rzeczywistości procesem tworzenia oprogramowania regułowego. Podobnie jak dla zwykłych języków programowania, wielce przydatne są tu narzędzia wspomagające ten proces.

Narzędzia takie powinny umożliwiać [7]:

- sprawdzanie poprawności reguł,
- identyfikację błędów w wykonaniu reguł,



Rys. 1. Moduł przeglądania reguł w trybie tekstowym

- przeglądanie zbioru reguł według ich powiązań,
- blokowanie i aktywację reguł.

System POSTGRES nie jest niestety wyposażony w tego typu narzędzia wspomagające. W celu usunięcia tego ewidentnego braku, w Instytucie Informatyki Politechniki Poznańskiej podjęto prace nad systemem **RDM – Rule Development and Maintenance System** [19]. RDM jest interakcyjnym systemem pracującym w środowisku Xwindows, wspomagającym programistę reguł systemu POSTGRES.

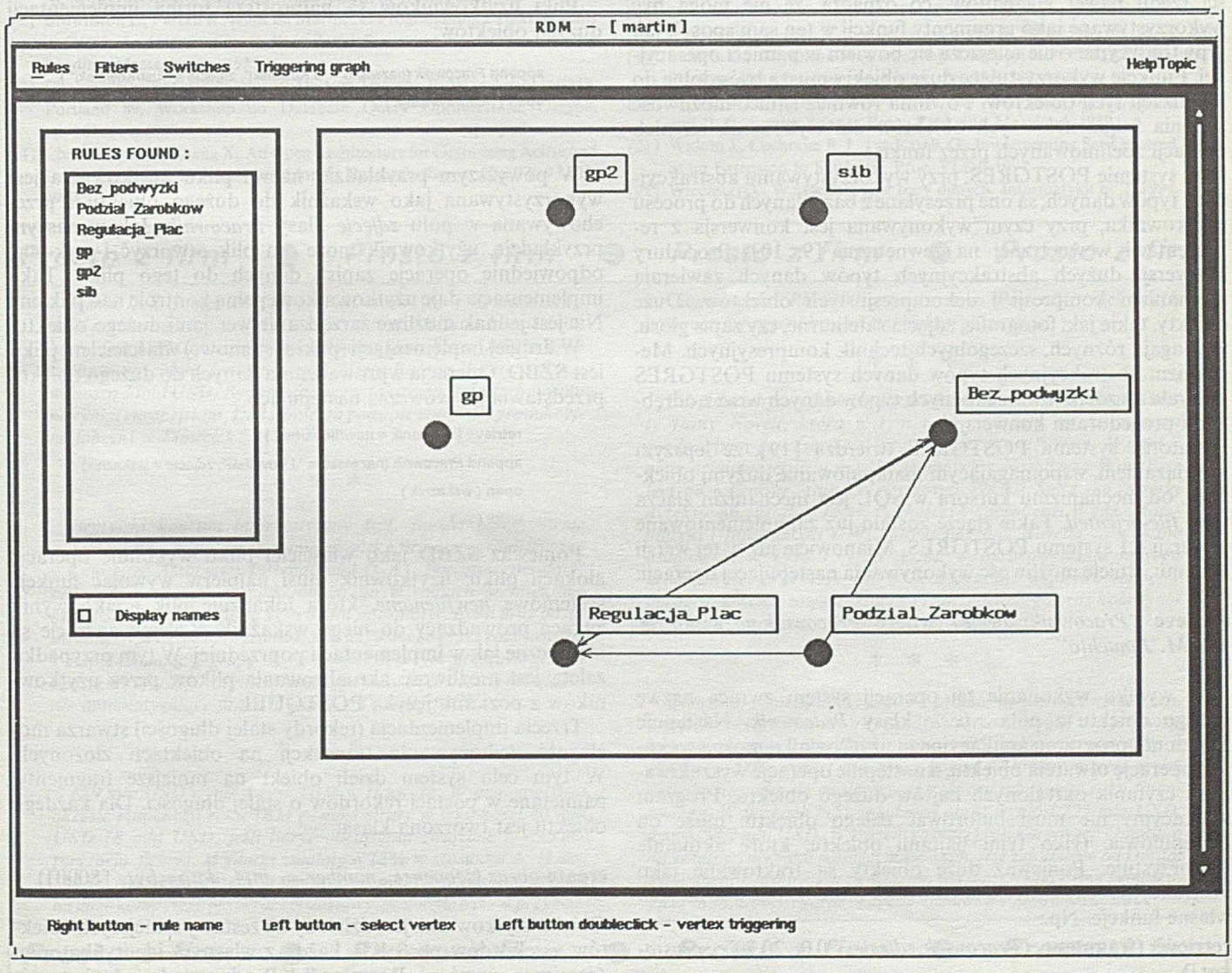
Zasadniczym narzędziem systemu RDM jest edytor reguł. Zamiast używać polecenia **define/remove rule**, użytkownik może, korzystając z edytora reguł, dokonać poprawki istniejącej reguły, wprowadzić nową lub usunąć istniejącą regułę. Gdy zbiór reguł jest duży, jego przeglądanie z wykorzystaniem polecenia **retrieve** może być utrudnione. Dla ułatwienia tego zadania RDM proponuje moduł przeglądania (ang. *browser*), który dodatkowo analizuje powiązania między regułami, wynikające z uruchamiania jednych reguł przez drugie. Jest to wygodne narzędzie analizy zależności również pod kątem możliwości wystąpienia cyklicznych wywołań. Uchwycenie tego niebezpieczeństwa jest istotne, gdyż cykliczne wywołania powodują nie tylko zawieszenie pracy systemu, ale mogą również doprowadzić do spowolnienia pracy całego systemu komputerowego. Przeglądanie można wykonywać w trybie tekstowym (lista reguł) lub w trybie graficznym (graf wywołań).

Przeglądanie bazy reguł w trybie tekstowym przedstawiono na rysunku 1. W lewym oknie jest wyświetlana lista reguł znalezionych w aktualnie przeglądanej aplikacji bazy danych. Wybranie jednej z reguł powoduje wyświetlenie informacji o innych regułach, które potencjalnie mogą być uaktywnione na podstawie warunku aktualnej reguły (patrz rys. 1 – TRIGGERED BY CONDITION) oraz informacji o klasach przeszukiwanych przy sprawdzeniu warunku. W kolejnym oknie są wyświetlane analogiczne informacje dla akcji reguły (patrz rys. 1 – TRIGGERED BY ACTION) i dodatkowo informacja o typie akcji – na rys. 1: REPLACE. Niżej prezentowany jest tekst reguły. Rysunek 2 przedstawia okno przeglądania grafu uruchomień reguł. Użytkownik może wybierać poszczególne reguły i badać potencjalne skutki ich uruchomienia.

W dużej bazie reguł często zachodzi potrzeba przeglądania bazy według określonych kryteriów [7]. Możemy tu wymienić wybór reguł o tym samym typie zdarzenia (np. **append**) lub odwołujących się do konkretnej klasy. W systemie POSTGRES reprezentacja reguły nie pozwala na łatwe formułowanie odpowiednich zapytań. System RDM rozwiązuje to przez analizę tekstu reguł. Dzięki utrzymywaniu przeanalizowanych definicji reguł, RDM umożliwia zakładanie filtrów, które przy przeglądaniu bazy reguł wybierają do prezentacji reguły według podanych kryteriów.

Manipulowanie obiektami złożonymi

Większość komercyjnych systemów baz danych dostarcza mechanizmów wspomagających z poziomu języka zapytań



Rys. 2. Graf wywołań reguł

przechowywanie i wyszukiwanie dużych obiektów binarnych (ang. *binary large objects* – BLOB). Na tym niestety kończą się możliwości manipulowania takimi obiektami. W systemach tych nie ma możliwości rozszerzenia języka zapytań o nowe, definiowane przez użytkowników funkcje i operatory, które mogłyby być przez nich wykorzystane do manipulowania dużymi obiektami. Co więcej, obiekty takie nie mogą być przekazywane do funkcji wewnętrznych systemu, a jedynie do programów aplikacyjnych użytkowników, w celu np. zbadania ich zawartości. Wyklucza to możliwość indeksowania wartości takich obiektów lub wyników programów aplikacyjnych na nich wykonywanych.

W systemie POSTGRES proponuje się traktowanie dużych, złożonych obiektów jako abstrakcyjnych typów danych ze złączem typu *file-oriented* i z różnymi formami implementacji. Abstrakcyjne typy danych systemu POSTGRES zostały omówione w pierwszej części niniejszego artykułu. Jak pokazano, z abstrakcyjnymi typami danych mogą być związane funkcje i operatory zdefiniowane przez użytkowników. Funkcje te, po ich zdefiniowaniu stają się integralną częścią systemu bazy danych i mogą być następnie wykorzystywane z poziomu systemu zarządzania bazą danych. Tą metodą możliwe jest również indeksowanie dużych obiektów.

Oczywiście problem manipulowania dużymi obiektami nie jest tak prosty, jak to mogłoby wynikać z przedstawionej wyżej charakterystyki [18, 19]. Nadal kilka problemów wymaga dalszych, intensywnych badań. Jednym z takich problemów jest kwestia rozmiarów dużych obiektów. Mogą one osiągać wielkości rzędu nawet gigabajtów, co oznacza, że nie mogą być wykorzystywane jako argumenty funkcji w ten sam sposób jak typy tradycyjne – nie mieszczą się bowiem w pamięci operacyjnej. Funkcje wykorzystujące duże obiekty muszą być zdolne do lokalizacji tych obiektów. Powinna również istnieć możliwość żądania fragmentów tych obiektów dla wykonania na nich operacji zdefiniowanych przez funkcje.

W systemie POSTGRES, przy wykorzystywaniu abstrakcyjnych typów danych, są one przesyłane z bazy danych do procesu użytkownika, przy czym wykonywana jest konwersja z reprezentacji wewnętrznej na zewnętrzną [9, 10]. Procedury konwersji dużych abstrakcyjnych typów danych zawierają mechanizm kompresji i dekompresji tych obiektów. Duże obiekty, takie jak: fotografie, zdjęcia satelitarne, czy zapis głosu, wymagają różnych, szczególnych technik kompresyjnych. Mechanizm abstrakcyjnych typów danych systemu POSTGRES pozwala na zdefiniowanie różnych typów danych wraz z odrębnymi procedurami konwersji.

Autorzy systemu POSTGRES twierdzą [19], że lepszym rozwiązaniem, wspomagającym manipulowanie dużymi obiektami, od mechanizmu kursora w SQL jest mechanizm złącza typu *file-oriented*. Takie złącze zostało już zaimplementowane w wersji 4.1 systemu POSTGRES. Mianowicie już w tej wersji systemu istnieje możliwość wykonywania następującej operacji:

retrieve *(Pracownik.zdjęcie)* **where** *Pracownik.nazwisko* = = *"M. Januchta"*

W wyniku wykonania tej operacji system zwraca nazwę dużego obiektu z pola *zdjęcie* klasy *Pracownik*. Następnie z poziomu programu aplikacyjnego użytkownika można wykonać operację otwarcia obiektu, a następnie operacje wyszukiwania i czytania określonych bajtów dużego obiektu. Program aplikacyjny nie musi buforować całego obiektu: może on manipulować tylko tymi bajtami obiektu, które aktualnie wykorzystuje. Ponieważ duże obiekty są traktowane jako abstrakcyjne typy danych, użytkownik może definiować na nich własne funkcje. Np.:

retrieve (fragment *(Pracownik.zdjęcie,* "0,0, 20,20":: *prostokąt))*

gdzie **fragment** jest funkcją abstrakcyjnego typu danych z dwoma argumentami: *zdjęcie* i *prostokąt* – funkcją, która wycina ze zdjęcia pracownika fragment zdefiniowany jako prostokąt. Podczas wykonywania tej funkcji jest tworzony tymczasowy, nowy duży obiekt. Jest on „wypełniany" bajtami z wykorzystaniem odpowiedniej sekwencji operacji **write**. Wskaźnik do tego obiektu jest zwracany przez funkcję. Tymczasowe duże obiekty są objęte mechanizmem *garbage-collection*, tak samo jak tymczasowe klasy po ich wykorzystaniu.

Składnia definicji dużego abstrakcyjnego typu danych jest nieco inna niż w przypadku typu konwencjonalnego, i przedstawia się następująco [10, 19]:

```
create large type nazwa_typu
        input = nazwa_procedury_1
        output = nazwa_procedury_2
        storage = forma_implementacji
```

gdzie **input** i **output** są procedurami konwersji reprezentacji zewnętrznej na wewnętrzną i odwrotnie, natomiast **storage** pokazuje formę implementacji dużego abstrakcyjnego typu danych. System POSTGRES umożliwia również definiowanie urządzenia, na którym obiekt ma być przechowywany (np. dysk twardy lub optyczny).

W systemie POSTGRES przewidziano cztery różne implementacje dużych obiektów [19], a mianowicie: pliki użytkowników, pliki systemowe, rekordy stałej długości oraz segmenty o zmiennej długości, zawierające dane użytkowników.

Pliki użytkowników są najprostszą formą implementacji dużych obiektów:

```
append Pracownik (nazwisko = "J.Kowalski", zdjecie = "/usr/Kowalski" )
open ( "/usr/Kowalski" )
write ( ... )
```

W powyższym przykładzie, nazwa pliku użytkownika jest wykorzystywana jako wskaźnik do dużego obiektu i przechowywana w polu *zdjęcie* klasy *Pracownik*. Jak w naszym przykładzie, użytkownik może ten plik otworzyć i wykonać odpowiednie operacje zapisu danych do tego pliku. Taka implementacja daje użytkownikowi pełną kontrolę nad plikiem. Nie jest jednak możliwe zarządzanie wersjami dużego obiektu.

W drugiej implementacji (pliki systemowe) właścicielem pliku jest SZBD. Operacja wprowadzania danych do dużego obiektu przedstawia się wówczas następująco:

```
retrieve ( wskaznik = newfilename ( ) )
append Pracownik (nazwisko = "J.Kowalski", zdjecie = wskaznik)
open ( wskaznik )
write ( ... )
```

Ponieważ SZBD jako właściciel pliku wykonuje operację alokacji pliku, użytkownik musi najpierw wywołać funkcję systemową *newfilename*, która lokalizuje plik i jako wynik zwraca prowadzący do niego wskaźnik. Kolejne operacje są identyczne jak w implementacji poprzedniej. W tym przypadku zaletą jest możliwość aktualizowania plików przez użytkowników z poziomu języka POSTQUEL.

Trzecia implementacja (rekordy stałej długości) stwarza możliwość wykonywania transakcji na obiektach złożonych. W tym celu system dzieli obiekt na mniejsze fragmenty, pamiętane w postaci rekordów o stałej długości. Dla każdego obiektu jest tworzona klasa:

create *obraz* (*sequence_number* = *int4, data*=*byte* [8000])

Obiekt użytkownika jest dzielony na zestawy mniejszych obiektów – rekordów po 8 KB, każdy z własnym identyfikatorem *(sequence_number)*. Rozmiar 8 KB odpowiada jednej stronie

systemu. Obiekt jest chroniony z wykorzystaniem mechanizmów systemowych; możliwe jest wykonywanie na nim transakcji oraz zapytań historycznych (ang. *time travel*). W systemie POSTGRES nie stosuje się techniki „nadpisywania" danych (ang. *overwrite data*). Przy odwoływaniu się do określonego rekordu najpierw jest wykonywana jego dekompresja (ang. *just-in-time conversion*). Zastosowane w systemie mechanizmy kompresji zmniejszają rozmiary przechowywanych danych o połowę.

Ostatnia implementacja stwarza możliwość przechowywania obiektu pod postacią zestawu segmentów o zmiennej długości. Segmenty te mogą być indeksowane:

segment_ndx *(locn, compressed_len, byte_pointer)*

Segmenty zmiennej długości są ze sobą połączone. Gdy obiekt jest powiększany, tworzy się nowy segment, a do indeksu jest dodawany nowy zapis pokazujący lokalizację segmentu na nośniku. Jednakże, ostateczną formą przechowywania obiektu są skompresowane wyżej opisane 8 KB bloki (choć jednostką kompresji są segmenty zmiennej długości, a nie jak poprzednio bloki 8 KB). Możliwe jest również wykonywanie zapytań historycznych.

LITERATURA

[1] Aiken A., Widom J., Hellerstein J. M.: Behaviour of database production rules: termination, confluence and observable determinism. In: Proc. of the ACM-SIGMOD Int. Conf. on Management of Data, June 1992

[2] Brownston L., Farrell R., Kant E., Martin N.: Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming. Addison-Wesley Reading, Massachusets, 1985

[3] Chakravarthy S.: Optimization of Alerters, Triggers, Rules in Active DBMS. Portland 89, Workshop on Database Query Optimization, Oregon, pp. 165–168, 1989

[4] Chakravarthy S., Zhang X.: An Open Architecture for Optimizing Active and Deductive Rules. University of Florida Research Report CIS-TR-93-013, 1993

[5] Hanson E. N.: Rule Condition Testing and Action Ececution in Ariel. Proc. ACM SIGMOD 1992, pp. 49–58

[6] Hanson E. N., Widom J.: An Overview of Production Rules in Database Systems. University of Florida Research Report CIS-TR-92-031, 1992

[7] Kotz-Dittrich A.: Adding Active functionality to an Object-Oriented Database System – a Layered Approach. Proc. Datenbanksysteme in Buro, Technik und Wissenschaft Conf., Braunschweig, Germany, March 1993

[8] McCarthy D. R., Dayal U.: The Architecture of an Active Data Base Management System. Proc. SIGMOD 1989, pp. 215–224

[9] The Postgres User Manual. Ed. J. Rhein, G. Kemnitz and the POSTGRES Group, EECS Dept. University of California, Berkeley, 1993

[10] The Postgres Reference Manual. Ed. POSTGRES Group, EECS Dept. University of California, Berkeley, 1993

[11] Rut M., Januchta M., Królikowski Z.: RDM: Tools for Rule Development and Maintenance for POSTGRES. Przedstawiono na: Intern. Conf. DEXA'94, Ateny, 1994

[12] Stonebraker M.: Triggers and Inference in Database Systems. In: On Knowledge Base Management Systems (Ed. M. Brodie, J. Mylopoulos), Springer-Verlag, 1986

[13] Stonebraker M. et al.: Extensibility in POSTGRES. IEEE Database Engineering, Sept. 1987

[14] Stonebraker M., Hanson E. N., Potamianos S.: The POSTGRES Rule Manager. IEEE Transactions on Software Engineering, vol. 14, No. 7, pp. 897–907, 1988

[15] Stonebraker M., Hearst M., Potamianos S.: A Commentary on the POSTGRES Rule System. ACM_SIGMOD Record, vol. 18, No. 3, 1989

[16] Stonebraker M., et al.: The Implementation of POSTGRES. IEEE Transactions on Knowledge and Data Engineering, March 1990

[17] Stonebraker M., Jhingran A., Goh J., Potamianos S.: On Rules, Procedures, Caching and Views Data Base Systems. In: Proc. of the ACM SIGMOD Intern. Conference on Management of Data, Atlantic City, New Jersey, pp. 281–290, 1990

[18] Stonebraker M., Kemnitz G.: The POSTGRES next-generation database management system. Communication of the ACM, 34 (10), pp. 78–92, 1991

[19] Stonebraker M., Olson M.: Large Object Support in POSTGRES. In: Intern. Conf. on Data Engineering, Vienna, pp. 355–362, 1993

[20] Ullman J. D.: Principles of Database and Knowledge Base Systems, Vol. I and II. Computer Science Press, Rockville, Maryland, 1989

[21] Widom J., Cochrane R. J., Lindsay B. G.: Implementing Set-Oriented Rules as an Extension to Starburst. Proc. VLDB 1991

[22] Wieczerzycki W.: Dedukcyjne bazy danych. Informatyka nr 2, 1994.

Prosto z firm ● Prosto z firm ● Prosto z firm ● Prosto z firm

Firma Dell Computer Corporation otworzyła w Texasie nowy dział marketingu i sprzedaży, w którym znajdzie zatrudnienie 1600 osób. Dla Della pracuje w jego głównej siedzibie w Austin 4700 osób, a na całym świecie ponad 7500. Roczne dochody firmy wynoszą 3 mld USD. By sprostać wzrastającemu zapotrzebowaniu na rynku europejskim, Dell zamierza podwoić zdolności produkcyjne fabryki w Limerick.

* * *

Brytyjski koncern informatyczny ICL zawarł porozumienie o strategicznej współpracy z amerykańską firmą programistyczną – Computer Associates International (CA). Porozumienie dotyczy przede wszystkim funkcjonującego w ramach architektury klient-serwer zarządzania systemami informatycznymi oraz bazami danych, a także oprogramowania używanego przez rozproszoną i współdziałającą grupę użytkowników, czyli groupware. CA będzie też oferował na emerykańskim rynku ICL-owskie rozwiązanie automatyzujące prace biurowe – TeamOFFICE.

* * *

689 mln USD wyniósł zysk IBM w II kw. br. W analogicznym okresie minionego roku IBM poniósł straty w wysokości 40 mln USD (8 mld USD, jeśli liczyć obciążenia związane z restrukturyzacją firmy). Wydatki zmalały o 18% w stosunku do II kw. 1993 r. Sprzedaż sprzętu przyniosła 7,7 mld USD obrotów. Popyt na duże komputery przekroczył dostawy, podobnie było w przypadku komputerów RS/6000 z procesorem PowerPC, niektórych komputerów przenośnych i największych pamięci.

* * *

250 polskich firm dystrybutorskich i dealerskich oferuje produkty firmy Novell, która w tym roku związała się ze znanym producentem procesorów tekstu – firmą WordPerfect. Wspieraniem działalności firm współpracujących z Novellem i WordPerfect od jesieni br. zajmuje się Novell Polska Sp. z o.o., której dyrektorem został p. Jacek Pacholczyk (absolwent Wydz. Matematyki i Informatyki UW, laureat nagrody Novella za wybitne osiągnięcia w 1993 r. i za udział w opracowywaniu nowego produktu w 1994 r.). Biuro będzie m.in. wspierać twórców oprogramowania towarzyszącego i narzędziowego oraz koordynować prace związane z lokalizacją produktów firmy.

* * *

Quantum – znany amerykański producent pamięci masowych, a przede wszystkim twardych dysków, odnotował za pierwszy kwartał swojego roku finansowego 1995, zakończonego 4 lipca 1994 r. – 51% wzrost obrotów i ponad 17-krotny wzrost zysku netto w stosunku do analogicznego okresu minionego roku finansowego. W liczbach bezwzględnych obrót wyniósł 725 mln USD, a zysk 58,2 mln USD. Wyrazem rosnącej roli korporacji Quantum na światowym rynku pamięci masowych, jest zakupienie od Digital Equipment Corporation za 400 mln USD części tej firmy produkującej m.in. dyski twarde, napędy taśmowe i głowice zapisu.

Prosto z firm ● Prosto z firm ● Prosto z firm ● Prosto z firm

Próba naśladowania natury – algorytmy genetyczne w informatyce

HALINA KWAŚNICKA
Centrum Informatyczne
Politechnika Wrocławska

Tematyka zastosowań algorytmów genetycznych GA (skrót od *Genetic Algorithms*) zdobywa szybko popularność. Ubiegłoroczny grudniowy numer AI EXPERT w całości jest poświęcony algorytmom genetycznym. Odbywają się międzynarodowe konferencje na ten temat. Powstają uniwersalne systemy opierające się na GA (C DARWIN II), duże systemy eksperckie (np. XpertRule) mają wbudowane GA do swoich modułów. Prowadzone są prace nad kombinacją GA i sieci neuronowych.

Ewolucja biologiczna jest bardzo skutecznym, naturalnym systemem adaptacyjnym. Wielu badaczy uważa, że te nadzwyczajne własności ewolucji naturalnej są przykładem godnym naśladowania. Organizmy żywe nabywają swoich zdolności przez dobór naturalny – pozornie ślepy mechanizm. Zasada doboru naturalnego – przeżywają i wydają potomstwo tylko jednostki najlepiej przystosowane – usuwa jedną z istotnych przeszkód w projektowaniu oprogramowania: konieczność uprzedniej specyfikacji wszystkich szczegółów problemu i wszystkich działań, które powinien podjąć program.

Popularność algorytmów genetycznych zaczęła wzrastać od połowy lat siedemdziesiątych, kiedy to John H. Holland opublikował znaną monografię *Adaptation in Natural and Artificial Systems* (*Ann Arbor: The University of Michigan Press,* 1975). Algorytmy genetyczne zaczęto stosować jako algorytmy przeszukujące (*in Search*), optymalizujące (*Optimization*) i uczące się (*Learning Machine*).

Programy wykorzystujące algorytmy genetyczne osiągają swój cel stosując strategię **doboru naturalnego.** Na początku program wytwarza początkową populację, tzn. możliwe rozwiązania. W analogii do żywych organizmów, różne rozwiązania wymieniają między sobą „geny" przez mechanizm *crossing over*, pojawiają się losowe zmiany w „genach" dzięki mechanizmowi mutacji. „Przeżywają" tylko rozwiązania najlepiej (lub stosunkowo dobrze) przystosowane, a ich „potomstwo" jest podstawą do szukania kolejnych rozwiązań.

Głównym tematem badaczy algorytmów genetycznych jest ich moc, równowaga między efektywnością a skutecznością, konieczna dla przeżycia w wielu różnych środowiskach. Możliwości zastosowań GA w przeszukiwaniach złożonych przestrzeni wykazano zarówno teoretycznie, jak i praktycznie. Zakres zastosowań algorytmów genetycznych jest coraz szerszy, ponieważ:

- obliczeniowo są proste,
- nie ma ostrych ograniczeń odnośnie przestrzeni poszukiwań (ciągłość, istnienie pochodnej, itp.),
- ich skuteczność jest zadowalająca.

Tradycyjne metody optymalizacji

Zadajmy sobie na wstępie pytanie: Czy konwencjonalne metody optymalizacyjne spełniają oczekiwania? W literaturze wyróżnia się trzy typy metod przeszukiwań (ang. *search methods*):

- obliczeniowe (ang. *calculus-based*),
- wyliczeniowe (ang. *enumerative*) i
- przypadkowe (ang. *random*).

Metody obliczeniowe są dokładnie zbadane. Dzielą się na dwie klasy: bezpośrednie (ang. *direct*) i pośrednie (ang. *indirect*). W metodach pośrednich szuka się lokalnego ekstremum przez rozwiązanie układu zwykle nieliniowych równań, otrzymanych z przyrównania gradientów funkcji do zera. Jest to uogólnienie szukania ekstremalnego punktu funkcji jednej zmiennej. Metody bezpośrednie szukają lokalnego optimum przez obliczanie wartości funkcji i przesuwanie się w kierunku lokalnego gradientu. Jest to sposób **osiągania szczytu** zwany *hill climbing*: znaleźć lokalnie najlepszy punkt, przesunąć się o zadany krok w obiecującym kierunku. Obie te klasy są badane i ulepszane, ale mimo to mają nadal istotne słabości.

Obie klasy metod działają na lokalnym obszarze: szukają najlepszego miejsca w sąsiedztwie aktualnego punktu. Jeśli nawet w miarę blisko będzie większe optimum (np. globalne), metody te mogą go nie wykryć, jeśli „po drodze" nastąpi pogorszenie wartości funkcji.

Metody te zależą również od istnienia pochodnych funkcji celu. Jeśli nawet dopuścilibyśmy numeryczną aproksymację pochodnych, niewiele to daje. Wiele praktycznych przestrzeni parametrów nie poddaje się takiej aproksymacji.

Wyliczeniowy schemat szukania optimum może być brany pod uwagę przy funkcjach o wielu kształtach i różnych rozmiarach. Idea jest bardzo prosta: w skończonej przestrzeni poszukiwań (parametrów) lub zdyskretyzowanej nieskończonej przestrzeni, algorytm przeszukiwania zaczyna działanie od wybranego punktu, sprawdzając wartości funkcji w kolejnych punktach przestrzeni parametrów, przy czym w danym momencie czasu jest sprawdzany jeden punkt przestrzeni. Prostota tego algorytmu jest bardzo atrakcyjna, natomiast jego efektywność jest cechą zniechęcającą, czasem wręcz uniemożliwiającą stosowanie. W wielu praktycznych problemach przestrzeń poszukiwań jest tak duża, że sprawdzanie punkt po punkcie (jeden w danym momencie) jest nie do przyjęcia ze względów czasowych. Nawet chwalone w literaturze programowanie dynamiczne (ang. *dynamic programming*) nie jest skuteczne w problemach o średnich rozmiarach i złożoności.

Algorytmy przeszukiwań losowych zaczęły zwiększać swą popularność, gdy badacze stwierdzili niedociągnięcia dwóch powyższych metod. Jednak losowa wędrówka po przestrzeni poszukiwań, połączona ze sprawdzaniem wartości funkcji i zachowywaniem najlepszego punktu, w wielu przypadkach nie jest zadowalająca z powodu jej małej efektywności. Przy długich przebiegach przeszukiwanie losowe nie wydaje się być lepsze od metody wyliczeniowej. Negując losowy algorytm przeszukiwania przestrzeni parametrów musimy wyraźnie stwierdzić, że nie jest to negacja losowych technik w ogóle. Inną, obecnie popularną techniką przeszukiwania jest tzw. *simulated annealing* – stosowanie procesów losowych do wprowadzenia procesu poszukiwań w stan minimalnej energii. W książce [2] analizowane są powiązania *simulated annealing* i *genetic algortihms*.

Pobieżny przegląd konwencjonalnych metod poszukiwania optymalnych rozwiązań wskazuje, że nie są one zadowalające w wielu praktycznych zagadnieniach, co nie oznacza, że są nieużyteczne. Zaró-

wno wspomniane wyżej metody, jak i ich niezliczone kombinacje mają zastosowania w wielu konkretnych problemach. Podsumowując, można następująco ocenić efektywność wyżej wspomnianych metod:

- Metody gradientowe są dobre dla wąskiej klasy problemów, lecz stają się bardzo mało użyteczne dla innych funkcji.
- Metody wyliczeniowe pod względem efektywności są podobne do przeszukiwań losowych. Wprawdzie nadają się dobrze dla szerokiego spektrum funkcji, lecz ich efektywność jest niezadowalająca.

Pożądana byłaby metoda o dość dużej efektywności dla szerokiej klasy problemów, którą nazwijmy „silna metoda". Możemy zgodzić się, że jej efektywność będzie wprawdzie mniejsza niż metod wyspecjalizowanych np. gradientowych dla określonych klas funkcji, jednak zawsze można próbować stworzenia metody kombinowanej z połączenia metod silnej i specjalistycznej dla danego problemu.

Zanim rozpatrzymy, na czym polega siła algorytmów genetycznych, przeanalizujmy, czym naprawdę jest **optymalizacja.** Teoria optymalizacji obejmuje analizę ilościowych metod poszukiwań optimum. Szuka się dokładnych sposobów opisu i osiągania tego, co jest lepsze, jeśli znana jest już miara tego, co jest dobre a co złe. Zatem optymalizacja jawi się jako proces ulepszania w kierunku określonego optymanego punktu (lub punktów).

Na optymalizację można patrzeć jako na dwie części: pierwszą – szukanie ulepszenia (polepszenia czegoś), drugą – sprawdzanie, czy punkt optymalny został osiągnięty. Przy takim podejściu widoczna jest różnica między **procesem ulepszania** a **osiągnięciem optimum.** W ocenie metod optymalizacji uwagę skupia się na ogół na zbieżności metody, tzn. stwierdzeniu czy metoda osiągnie optimum, zapominając o jej działaniu pośrednim. Podejście takie nie wydaje się być naturalne i wynika z pierwszych metod obliczeniowych. Aby się o tym przekonać, wystarczy zwrócić uwagę na to, jak oceniane są decyzje człowieka, np. zarządzającego przedsiębiorstwem? Z reguły ocena jest wtedy dobra, jeśli człowiek podejmuje właściwą decyzję spośród możliwych lub dostępnych w danym czasie. Nie oceniamy szefa przedsiębiorstwa wyłącznie pod kątem, czy osiągnie on globalnie najlepszy wynik. „Dobroć" jego decyzji jest oceną relatywną w stosunku do innych, możliwych w tym czasie, decyzji. Jesteśmy zainteresowani, by robić lepiej niż inni. Jeśli zatem chcielibyśmy narzędzia optymalizującego, zbliżonego do sposobów działania człowieka, to musielibyśmy przeorientować priorytety optymalizacji.

Najważniejszym celem optymalizacji jest stałe polepszanie. Czy możliwe jest jednak szybkie osiągnięcie zadowalającego poziomu „dobroci" rozwiązania? Samo osiągnięcie optimum w złożonych systemach jest mniej ważne. Dobrze byłoby również osiągnąć optimum szybko, ale nas zadowolić może tylko szybkie znajdowanie coraz lepszych rozwiązań, w konsekwencji szybkie znalezienie rozwiązania satysfakcjonującego.

Działanie algorytmów genetycznych

Wyróżnić można cztery podstawowe różnice między algorytmami genetycznymi a tradycyjnymi procedurami poszukiwania rozwiązań optymalnych [4]:

- Działają z zakodowanym zbiorem parametrów, nie bezpośrednio z parametrami.
- Poszukują, działając jednocześnie z populacją punktów, a nie z pojedynczym punktem w przestrzeni parametrów.
- Nie posługują się pochodnymi funkcji ani też inną wiedzą o funkcji celu.
- Stosują probabilistyczne reguły przechodzenia po przestrzeni parametrów.

Algorytmy genetyczne wymagają, aby zbiór parametrów w problemie optymalizacji był kodowany na łańcuchy skończone długości, w określonym skończonym alfabecie. Każdy łańcuch reprezentuje jeden punkt w przestrzeni parametrów. Populacja łańcuchów reprezentuje zbiór testowanych punktów. W każdej chwili znana jest tylko wartość funkcji w aktualnie testowanych punktach przestrzeni parametrów, tzn. „jakość" danego łańcucha. Preferowanie „lepszych" łańcuchów i ich powielanie z możliwymi losowymi zmianami łańcuchów, powoduje powstanie nowych łańcuchów, a zatem określenie nowych punktów w przestrzeni parametrów, które są testowane w kolejnej chwili czasu (generacji populacji). Ponieważ algorytmy genetyczne „przeskakują" z wielu punktów w przestrzeni parametrów do wielu punktów, większe są szanse na to, iż zostaje znalezione globalne optimum, a algorytm nie zatrzyma się na pierwszym znalezionym, niekoniecznie globalnym optimum. Możliwe jest znalezienie kilku optimów jednocześnie: różne szczyty mogą być znalezione przez różne łańcuchy w tej samej populacji.

GA stosują losowe reguły przechodzenia w przestrzeni parametrów (losowe tworzenie łańcuchów w kolejnych populacjach), co nie oznacza jednak, że jest to losowe szukanie (przypadkowe „błądzenie"). GA preferując „lepsze" łańcuchy, stosuje losowy wybór jako narzędzie wprowadzenia poszukiwań w te regiony przestrzeni parametrów, które są lepsze.

Ogólny schemat działania algorytmów genetycznych

Poszukiwanie optymalnego (w języku algorytmów genetycznych) rozwiązania polega na znalezieniu określonego łańcucha – genotypu osobnika najlepiej przystosowanego (tzn. o największej wartości funkcji optymalizacyjnej, nazywanej funkcją przystosowania).

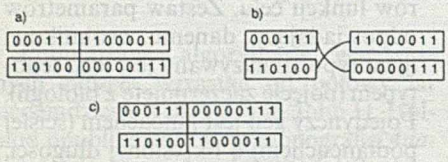
Aby rozwiązać zadanie za pomocą prostego algorytmu genetycznego, należy wykonać następujące czynności:

- Kodowanie parametrów funkcji na łańcuchy o skończonej długości. Jest to definiowanie genotypu elementów populacji.
- Ustalenie parametrów algorytmu: wielkości populacji i parametrów operatorów genetycznych.
  Każdy osobnik (element) w populacji jest zestawem genów – „genotypem", w najprostszym przypadku geny są umieszczone na pojedynczym łańcuchu – „chromosomie". Jeden gen jest odwzorowaniem wartości jednego parametru. Zatem osobnik jest odwzorowaniem zestawu parametrów funkcji, czyli punktem w przestrzeni parametrów funkcji celu. Zestaw parametrów odpowiadający danemu osobnikowi (genotypowi) nazywany jest jego fenotypem (pojęcie zaczerpnięte z biologii). Pojedynczy gen jest łańcuchem (ściślej podłańcuchem) o określonej długości, składającym się z symboli wybranego alfabetu; najczęściej jest przyjmowany alfabet binarny – zera i jedynki, choć może też być inny alfabet.
- Ustalenie (najczęściej losowe) początkowej populacji łańcuchów (początkowa populacja elementów).
- Cykliczne generowanie kolejnych populacji łańcuchów stosując algorytm genetyczny do chwili, aż pojawi się warunek zatrzymania ewolucji (warunkiem zatrzymania może być zadana liczba generacji, liczba generacji w czasie których nie znaleziono lepszego rozwiązania itp.). Na generowanie kolejnej generacji populacji składają się:
  A. Mechanizm selekcji: wybór elementów z populacji do procesu reprodukcji. W poszczególnych implementacjach wybór ten może być różny. Na ogół większe szanse wyboru ma osobnik lepiej przystosowany, tzn. o większej niż pozostałe wartości funkcji przystosowania. W zależności od problemu, funkcją przystosowania może być zysk przedsiębiorstwa, koszt wytworzenia wyrobu, długość drogi (w przypadku problemu komiwojażera) lub podana analityczna postać funkcji.

Każdy element jest indywidualnie oceniany pod względem przydatności strategii, jaką koduje.

B. Reprodukcja elementów: jest to proces powielania elementów, tworzenia elementów potomnych (potomków), które wejdą w skład populacji w następnej generacji (następnym pokoleniu). Liczba potomków każdego elementu zależy od jego przystosowania. W procesie reprodukcji występują dwa operatory ewolucyjne, powodujące powstanie różnorodności w populacji: procesy **crossover** i **mutacji.**

– Spośród wybranych do reprodukcji elementów wybiera się (losowo) po dwa łańcuchy „rodzicielskie" do przeprowadzenia operacji *crossing-over*. Na jednym z łańcuchów „rodzicielskich" wybiera się losowo jeden punkt, po czy zamienia się części łańcuchów rodzicielskich leżące po jednej stronie wybranego punktu (jest to *crossing-over* pojedyncze, można stosować też wielokrotne). W rezultacie tego zabiegu otrzymuje się dwa elementy „potomne". Na rys. pokazano schemat *crossing-over*, przy czym do reprezentacji parametrów (jako wartości genów) wybrano alfabet binarny – zera i jedynki.



Schemat działania operatora crossing-over
a) elementy „rodzicielskie",
b) cross-over,
c) elementy „potomne"

Crossing-over jest podstawowym operatorem mieszającym geny, zarówno w algorytmach genetycznych, jak i u żywych organizmów. W tym miejscu algorytm genetyczny ściśle naśladuje swój biologiczny pierwowzór, ponieważ u organizmów żywych chromosomy (łańcuchy genów) podlegają *crossing-over* w momencie tworzenia zygoty (zlewania się gamet). W wyniku *crossing-over* nastąpi nie tylko przemieszanie cech, może też powstać nowa cecha (jeśli przełamanie chromosomu nastąpi w środku genu).

– Powstałe potomsto jest poddawane działaniu kolejnego operatora genetycznego – mutacji. **Mutacja** jest losową zmianą genu (elementu łańcucha). Mutacje same zwykle nie wystarczają do znalezienia optymalnego rozwiązania, ale przyczyniają się do istnienia różnorodności w populacji oraz nie dopuszczają do wytworzenia się populacji jednorodnej, co uniemożliwiłoby dalszą jej ewolucję. Zazwyczaj mutacji podlega niewielka część populacji.

C. Wybór elementów do następnej generacji (nowej populacji). W niektórych implementacjach stosuje się wybór najlepszych elementów (ich liczba określona jest przez parametr **rozmiar populacji**) spośród potomków (cała populacja „wymiera", „żyje" tylko najlepsza część potomstwa); w innych implementacjach – do następnej generacji są wybierane najlepsze elementy spośród zbioru „rodziców" i „potomków". Czasami jest stosowana strategia pośrednia: do następnej generacji bezwarunkowo „przeżywa" najlepszy element z populacji „rodziców", resztę stanowi „potomstwo".

W niektórych implementacjach GA stosuje się dodatkowe, bardziej zaawansowane mechanizmy, operujące na poziomie genotypu (zaczerpnięte z genetyki): istnienie genów dominujących, inwersję, wewnątrzchromosomową duplikację, translokację i segregację.

Geny dominujące – mają znaczenie u elementów diploidalnych w odwzorowaniu genotypu na fenotyp. Każdy gen może mieć dwa allele, które oznaczmy jako allel A oraz allel a. Osobnik zawiera dwa chromosomy homologiczne, z których każdy może zawierać allel A lub a. Jeśli założymy, że dominujący dla tego genu jest allel A, to allel a może mieć wpływ na fenotyp tylko u osobników mających zestaw alleli aa, natomiast w zestawach AA, Aa, aA na fenotyp osobnika będzie wpływał allel A [4].

Inwersja (oraz inne operatory zmieniające kolejność) – wygląda na to, że natura sama nie jest pewna, czy jej sposób kodowania jest właściwy i wynalazła operatory, które w tym samym czasie, gdy następują poszukiwania lepszego zbioru wartości alleli, szukają lepszego kodowania. Są to operatory zmieniające ustawienie genów w chromosomach (ang. *reordering operators*).

Operator inwersji – wybierane są dwa punkty na chromosomie i część między tymi punktami jest odwracana (ustawiana w odwrotnej kolejności). Biorąc pod uwagę rozszerzoną reprezentację, w której dany allel znaczy to samo, niezależnie od miejsca w łańcuchu, w którym występuje (tak jest w naturze), to prosta inwersja nie ma efektu w odkodowanej wartości parametru. Dlaczego zatem istnieje w biologii? Inwersja może być użyteczna dla szukania dobrego rozmieszczenia w łańcuchu w czasie, gdy działają inne operatory szukające dobrego zbioru alleli (w [4] – s. 166–179). Mechanizm ten powoduje od czasu do czasu przetasowanie genów w ten sposób, że geny odległe u rodziców stają się genami bliskimi u potomka. Systemy takie potrafią wykryć i faworyzować bardziej zwarte schematy (pojęcie schematu wyjaśnione jest w rozdziale „Na czym polega siła algorytmów genetycznych?").

Operator translokacji – może być widziany jako międzychromosomowy *crossing-over*. Wpływa on na organizację chromosomu. Aby go wprowadzić, należy związać allele z nazwą ich genu tak, by można było identyfikować ich właściwe znaczenie, gdy są one pomieszane na chromosomie (w [4] – s. 179–180).

Operator segregacji (działa w koniunkcji z operatorem translokacji) – zakłada się, że populacja składa się z osobników nie diploidalnych (dwa chromosomy homologiczne), lecz poliploidalnych (wiele chromosomów homologicznych). Spójrzmy na proces tworzenia gamety, gdy mamy do czynienia z więcej niż jedną parą chromosomów homologicznych. *Crossing-over* działa jak poprzednio, tyle że losowo wybierany jest każdy z haploidalnych chromosomów. Ten proces losowej selekcji jest nazywany segregacją (ang. *segregation*). Segregacja jest użyteczną, jeśli stosunkowo niezależne geny będą przypadkowo ulokowane na różnych chromosomach. Dzięki temu złe allele nie będą mogły przeżyć przez „więzienie się" na wysoko przystosowanych nieskorelowanych allelach.

Operatory duplikacji i delecji – para operatorów niskiego poziomu wprowadzana w sztucznych GA. Holland sugeruje, że mogą one być efektywną metodą dla sterowania współczynnikiem mutacji. Duplikacja polega na zduplikowaniu poszczególnego genu i umieszczeniu go obok swego „przodka". Delecja jest to usunięcie z chromosomu zduplikowanego genu.

Wyróżnienie płci w populacji uważa się za czynnik mało znaczący w sztucznych algorytmach genetycznych, stosowanych do celów optymalizacyjnych. Uwzględnianie płci w strukturze populacji może mieć znaczenie przy badaniu procesów ewolucyjnych, w których interesować nas może szybkość i trasa wędrówki populacji w krajobrazie przystosowawczym, możliwość „przeskoczenia" z jednego szczytu przystosowawczego do innego itp.

Opis formalny algorytmów genetycznych

Algorytm genetyczny może być opisany jako ósemka [9]:

$$GA = (P^0, m, l, s, \rho, \Omega, f, T),$$

gdzie $P^0 = (e_1^0, \ldots, e_m^0) \in I^m$, $I = \{0,1\}^l$ – populacja początkowa,
$m \in N$ – rozmiar populacji (liczba osobników w populacji),

Spis treści rocznika INFORMATYKA 1994

Artykuły problemowe

	nr	s.
ABRAMOWICZ WITOLD: Akademicka informatyka ekonomiczna w krajach niemieckojęzycznych	7	18
AUKSZTOL JERZY: Jakość oprogramowania	12	17
BASZTURA CZESŁAW: AWEK – Akustyczne wejście do komputera	12	20
BEM DANIEL, JANUKIEWICZ JAROSŁAW M.: Czy już nadszedł czas na ATM?	10	8
BIEŃKO ZYGMUNT: Zastosowanie standardów EDIFACT w statystyce	4	15
BURAKOWSKI WOJCIECH: Obsługa sieci LAN-LAN w sieci B-ISDN ATM	10	27
CZAJKOWSKI GRZEGORZ, ZIELIŃSKI KRZYSZTOF: PCN – system rozproszonego przetwarzania	5	8
CZARNECKA-CHROBOT BEATA: Metoda FUNCTION POINTS	5	19
DANIKIEWICZ EUGENIUSZ: Standard UN/EDIFACT	1	11
DZIUBA DARIUSZ, SCHEER A.W., ELSNER T., LASEK MIROSŁAWA: ARIS – Architektura Zintegrowanych Systemów Informatycznych	9	8
ELSNER T., SCHEER A.W., DZIUBA DARIUSZ, LASEK MIROSŁAWA: ARIS – Architektura Zintegorwanych Systemów Inforamatycznych	9	8
FILIPIAK JANUSZ, LASOŃ ARTUR, PACH ANDRZEJ R.: Szerokopasmowe sieci miejskie zgodne ze standardem IEEE 802.6 DQDB	10	10
FRYŹLEWICZ ZBIGNIEW, UŁASIEWICZ JĘDRZEJ: System operacyjny QNX jako urządzenie tworzenia aplikacji czasu rzeczywistego	12	22
GAJĘCKI MAREK, ZIELIŃSKI KRZYSZTOF: PVM – środowisko do przetwarzania rozproszonego w sieciach stacji roboczych	6	19
GŁÓWKA ZBIGNIEW: Cieszyński Model Celny	4	18
GOMÓŁKA ZDZISŁAW: Informatyka ekonomiczna w programie kszalcenia ekonomistów	7	16
GÓRSKI JANUSZ, WARDZIŃSKI ANDRZEJ: Analiza bezpieczeństwa komputerowego systemu sterowania	8	7
GREGA WOJCIECH, KOŁEK KRZYSZTOF: Ekonomiczne realizacje układów sterowania czasu rzeczywistego na IBM PC	4	1
GRZEGORZEWSKI MARCIN, RZESIOWSKI DARIUSZ: Sieci neuronowe i logika rozmyta w systemach eksperckich	6	11
HADYNIAK BOGUSŁAW M.: Problemy zastosowań EDI	1	7
HAŁAS ELŻBIETA: EANCOM – praktyczny przewodnik wdrażania zasad UN/EDIFACT w handlu i przemyśle	2	25
HAŁKA WOJCIECH: Perspektywy rozwoju krajowej sieci telekomunikacynej dla EDI	3	20
HOFMOKL TOMASZ, WĘGLARZ JAN, WIERZBICKI ANDRZEJ P.: Miejskie sieci komputerowe w infrastrukturze informatycznej nauki w Polsce	10	5
HOŁODNIK-JANCZURA GRAŻYNA, PŁONKA-SZYDLAK ELŻBIETA, ZABAWSKA EWA: Realizacja przedsięwzięć informatycznych z wykorzystaniem metodyki CASE firmy Oracle	2	14
HUDYMA ELŻBIETA: Testowanie oprogramowania	8	22
HUZAR ZBIGNIEW: Projektowanie oprogramowania systemów czasu rzeczywistego. Podstawowe fazy projektu	8	11
HUZAR ZBIGNIEW, MAGOTT JAN: Systemy czasu rzeczywistego – metody specyfikacji	8	14
JANUCHTA MARCIN, KRÓLIKOWSKI ZBYSZKO: POSTGRES – rozszerzalny system zarządzania bazą danych	11	7
JANUCHTA MARCIN, KRÓLIKOWSKI ZBYSZKO: POSTGRES jako narzędzie budowy aktywnych i dedukcyjnych systemów baz danych	12	4
JANUKIEWICZ JAROSŁAW M., BEM DANIEL: Czy już nadszedł czas na ATM?	10	8
JOHNSON CLARE, SAPTE WILDE: Aspekty prawne EDI	4	26
KACZYŃSKI JAROSŁAW A.: Nowoczesne symulatory układów elektronicznych dla MS-Windows	6	10

	nr	s.
KOCIĘCKI MACIEJ: POLPAK – publiczna sieć pakietowej transmisji danych	3	17
KOLBUSZ EDWARD, OLEJNICZAK WOJCIECH: Problemy informatyki ekonomicznej	7	2
KOŁEK KRZYSZTOF, GREGA WOJCIECH: Ekonomiczne realizacje układów sterowania czasu rzeczywistego na IBM PC	4	1
KOSMACZ-CHODOROWSKA ANNA: Kody kreskowe i automatyczna identyfikacja dla potrzeb EDI	2	21
KOSMULSKA-BOCHENEK ELŻBIETA: Synchronizacja w rozproszonych systemach uwarunkowanych czasowo	11	17
KRÓLIKOWSKI ZBYSZKO, JANUCHTA MARCIN: POSTGRES – rozszerzalny system zarządzania bazą danych	11	7
KRÓLIKOWSKI ZBYSZKO, JANUCHTA MARCIN: POSTGRES jako narzędzie budowy aktywnych i dedukcyjnych systemów baz danych	12	4
KURAŚ MARIAN: Kształcenie analityków systemów informacyjnych	7	14
KWAŚNICKA HALINA, MARKOWSKA-KACZMAR URSZULA: Sztuczna inteligencja w sterowaniu	6	3
KWAŚNICKA HALINA, MARKOWSKA-KACZMAR URSZULA: Zastosowanie sieci neuronowych w systemach czasu rzeczywistego	11	3
KWAŚNICKA HALINA: Próba naśladowania natury – algorytmy genetyczne w informatyce	12	12
LANGE KATARZYNA: Podstawowe pojęcia EDI	1	16
LASEK MIROSŁAWA, SCHEER A.W., ELSNER T., DZIUBA DARIUSZ: ARIS – Architektura Zintegrowanych Systemów Informatycznych	9	8
LASOŃ ARTUR, FILIPIAK JANUSZ, PACH ANDRZEJ R.: Szerokopasmowe sieci miejskie zgodne ze standardem IEEE 802.6 DQDB	10	10
LEWANDOWSKI WITOLD D.: Prywatne sieci telekomunikacyjne	3	23
ŁAWRYNOWICZ ANNA: Ekspercki system wspomagania sterowania produkcją	6	27
MAGOTT JAN, HUZAR ZBIGNIEW: Systemy czasu rzeczywistego – metody specyfikacji	8	14
MARKOWSKA-KACZMAR URSZULA, KWAŚNICKA HALINA: Sztuczna inteligencja w sterowaniu	6	3
MARKOWSKA-KACZMAR URSZULA, KWAŚNICKA HALINA: Zastosowanie sieci neuronowych w systemach czasu rzeczywistego	11	3
METELKIN JOANNA: MegaCAD – prosty program dla wymagających	6	18
MIŁOSZ MAREK, PRANAGAL ROBERT: Budowa prototypu systemu ekspertowego	5	14
MRÓZEK ADAM, WINIARCZYK RYSZARD: Systemy operacyjne czasu rzeczywistego	9	23
NIEDZIELSKA ELŻBIETA: Zmierzch czy nowa epoka rozwoju akademickiej informatyki ekonomicznej	7	5
NIEDŹWIEDZIŃSKA HANNA: Badania naukowe i nauczanie w zakresie EDI	3	22
NOWAKOWSKI ANTONI: Informatyka jako narzędzie ekonomisty – określenie celów kształcenia	7	13
OLEJNICZAK WOJCIECH, KOLBUSZ EDWARD: Problemy informatyki ekonomicznej	7	2
PACH ANDRZEJ R., LASOŃ ARTUR, FILIPIAK JANUSZ: Szerokopasmowe sieci miejskie zgodne ze standardem IEEE 802.6 DQDB	10	10
PŁONKA SZYDLAK ELŻBIETA, HOŁODNIK-JANCZURA GRAŻYNA, ZABAWSKA EWA: Realizacja przedsięwzięć informatycznych z wykorzystaniem metodyki CASE firmy Oracle	2	14
POTEMPA ANDRZEJ: System CAD/CAM/CAE UNIGRAPHIC	6	23
PRANAGAL ROBERT, MIŁOSZ MAREK: Budowa prototypu systemu ekspertowego	5	14
PRUCHNIK RADOSŁAW: Odwracalna kompresja za pomocą kodowania arytmetycznego	3	11
RUSZNIAK RYSZARD: Zastosowanie symulacji dyskretnej do badania dynamiki systemów informacyjnych organizacji gospodarczych	7	26
RYBIŃSKI WITOLD, WRYCZA STANISŁAW: Nauczanie systemów informatycznych. Założenia i praktyka	7	22
RZESIOWSKI DARIUSZ, GRZEGORZEWSKI MARCIN: Sieci neuronowe i logika rozmyta w systemach eksperckich	6	11
SACHA KRZYSZTOF: Standardy planowania jakości, dokumentacji i konserwacji oprogramowania	8	18
SALAMOŃCZYK JAN: PLODETTE dla polskiego przemysłu motoryzacyjnego	4	25
SAPTE WILDE, JOHNSON CLARE: Aspekty prawne EDI	4	26
SCHEER A.W., ELSNER T., DZIUBA DARIUSZ, LASEK MIROSŁAWA: ARIS – Architektura Zintegrowanych Systemów Informatycznych	9	8

	nr	s.
SHEPHERD ALAN: Zastosowanie EDI w zachodnioeuropejskim przemyśle samochodowym	4	22
SIENKIEWICZ PIOTR: Dylematy informatycznego kszalcenia	7	11
SIMIŃSKI ROMAN: Projekt i realizacja obiektowo zorientowanego podsystemu komunikacji z użytkownikiem	2	8
SROKA HENRYK: Wybrane problemy kszalcenia w zakresie informatyki ekonomicznej	7	17
STABROWSKI MAREK: Podprogramy w języku symulacji systemów dynamicznych	9	16
STOCHLAK JACEK: Systemy przetwarzania transakcyjnego OLTP	3	1
STOKŁOSA JANUSZ: Jeszcze raz o kluczach szyfrujących w metodzie RSA	1	26
STOKŁOSA JANUSZ: O standaryzacji w zakresie technik kryptograficznych	9	13
STROIŃSKI MACIEJ: Technologia miejskich sieci komputerowych z perspektywy użytkownika i operatora sieci	10	21
SZEWCZYK AGNIESZKA: Rozwój systemu informacyjnego jako rezultat działań diagnostycznych i odnawiających	4	10
SZYJEWSKI ZDZISŁAW: Środowisko stosowania narzędzi CASE	1	21
SZYJEWSKI ZDZISŁAW: Podatkiem w naukę	7	20
SZYMAŃSKI ZBIGNIEW: Perspektywy rozwoju transmisji światłowodowej	10	16
TYMOWSKI JAROSŁAW: Elektroniczna Izba Rozliczeniowa	4	14
UŁASIEWICZ JĘDRZEJ, FRYŹLEWICZ ZBIGNIEW: System operacyjny QNX jako narzędzie tworzenia aplikacji czasu rzeczywistego	12	22
WARDZIŃSKI ANDRZEJ, GÓRSKI JANUSZ: Analiza bezpieczeństwa komputerowego systemu sterowania	8	7
WĘGLARZ JAN, HOFMOKL TOMASZ, WIERZBICKI ANDRZEJ P.: Miejskie sieci komputerowe w infrastrukturze informatycznej nauki w Polsce	10	5
WIECZERZYCKI WALDEMAR: Dedukcyjne bazy danych	2	2
WIERZBICKI ANDRZEJ P., HOFMOKL TOMASZ, WĘGLARZ JAN: Miejskie sieci komputerowe w infrastrukturze informatycznej nauki w Polsce	10	5
WINIARCZYK RYSZARD, MRÓZEK ADAM: Systemy operacyjne czasu rzeczywistego	9	23
WITKOWSKI TADEUSZ: Metody zbiorów rozmytych w analizie decyzyjnej przedsiębiorstwa.	nr	s.
Część 1	8	26
Część 2	9	20
WOJDAŁA ANDRZEJ: Architektura otwartego systemu do syntezy obrazów realistycznych	9	3
WRYCZA STANISŁAW, RYBIŃSKI WITOLD: Nauczanie systemów informatycznych. Założenia i praktyka	7	22
ZABAWSKA EWA, HOŁODNIK-JANCZURA GRAŻYNA, PŁONKA-SZYDLAK ELŻBIETA: Realizacja przedsięwzięć informatycznych z wykorzystaniem metodyki CASE firmy Oracle	2	14
ZIELIŃSKI KRZYSZTOF, CZAJKOWSKI GRZEGORZ: PCN – System rozproszonego przetwarzania	5	8
ZIELIŃSKI KRZYSZTOF, GAJĘCKI MAREK: PVM – środowisko do przetwarzania rozproszonego w sieciach stacji roboczych	6	19
ŻURAKOWSKI ZDZISŁAW: Niektóre zagadnienia związane z projektowaniem i eksploatacją oprogramowania na poziomie profesjonalnym	8	4

Wywiady

Budzenie świadomości środowiska – z dr. Romanem Dolczewskim rozmawia *Krystyna Karwicka-Rychlewicz*	5	1
Współpraca konkurentów – z dr. Wacławem Iszkowskim – rozmawia *Krystyna Karwicka-Rychlewicz*	6	1
O kształceniu analityków systemów – z prof. dr. hab. inż. Janem Golińskim rozmawia *Krystyna Karwicka-Rychlewicz*	7	9
Bez uszczęśliwiania na siłę – z Piotrem Fuglewiczem rozmawia *Krystyna Karwicka-Rychlewicz*	8	1
Edukacja użytkownika – z dr. Andrzejem Florczykiem rozmawia *Krystyna Karwicka-Rychlewicz*	9	1
Między NASK i LAN – z prof. dr. hab. inż. Janem Węglarzem rozmawia *Krystyna Karwicka-Rychlewicz*	10	3
Szanse i zagrożenia – z Teresą Możdzyńską – wiceprezesem Głównego Urzędu Statystycznego rozmawiają *Krystyna Karwicka* i *Władysław Klepacz*	11	2
PESEL – zbiór podstawowy – z Dariuszem Kupieckim, dyrektorem Rządowego Centrum Informatycznego PESEL rozmawia *Krystyna Karwicka-Rychlewicz*	12	2

Systemy czasu rzeczywistego

Systemy operacyjne czasu rzeczywistego – *Adam Mrózek, Ryszard Winiarczyk*	9	23
Synchronizacja w rozproszonych systemach uwarunkowanych czasowo – *Elżbieta Kosmulska-Bochenek*	11	17

	nr	s.
System operacyjny QNX jako narzędzie tworzenia aplikacji czasu rzeczywistego – *Zbigniew Fryźlewicz, Jędrzej Ułasiewicz*	12	

Polemiki

Uwagi na temat kluczy szyfrujących w metodzie RSA – *Zygmunt Topolewski*	7	29
Odpowiedź na polemikę „Co to jest uniwersytet? Andrzeja Horodeńskiego, zamieszczoną w nr. 6/94 ENTER – *Kazimierz Subieta*	8	31

Ankiety

Ankieta sieciowa	5	27
Ankieta sieciowa (omówienie) – oprac. *Krystyna Karwicka*	11	14

Z kraju

Trzydziestolecie INFORMATYKI	1	2
I Kongres Informatyki Polskiej	7	1
Oracle przekazuje oprogramowanie polskim uczelniom	7	10
Wyniki X Ogólnopolskiego Konkursu na najlepsze prace magisterskie z informatyki	7	25
CRAY Y/MP EL98 w Warszawie – (*WBI*)	7	30
Są wśród nas	8	6
I Kongres Informatyki Polskiej	9	2
I Kongres Informatyki Polskiej. Analiza środowiska informatycznego	10	1
Łańsk '94	10	2
I Kongres Informatyki Polskiej	11	1
Sequent i Unisys rozszerzają porozumieie OEM – (*mw*)	11	15
SOFTARG '94 – *Władysław Klepacz*	12	28

Ze świata

Targi CeBIT '94 – *WK*	1	III okł.
Główni wytwórcy komputerów osobistych – *Jan Ryżko*	5	25
CeBIT '94 – Refleksje – *Władysław Klepacz*	5	28
Kongres IFIP '94 – *WK*	8	2
Największe firmy informatyczne w 1993 roku – *Jan Ryżko*	11	22

Konferencje

Elektroniczna wymiana danych– *Marian Niedźwiedziński*	1	3
3. Krajowa konferencja: Komputery w medycynie. 23–24 czerwca 1994 r.	1	20
8. Międzynarodowe Sympozjum: SYSTEM-MODELLING – CONTROL. 1–5 maja 1995 r.	1	25
Wiosenna Szkoła PTI. Języki 4GL	2	III okł.
INFOSYSTEM '94. 17–20 maja 1994 r.	2	III okł.
Szósta Świętojańska Wyższa Międzynarodowa Wdrożeniowa Górska Szkoła PTI Szczyrk. 20–24 czerwca 1994 r.	3	IV okł.
Miejskie Sieci Komputerowe w nauce i gospodarce POLMAN '94. 16–17 maja 1994 r.	4	IV okł.
Wiosenna Szkoła PTI '94. 16–20 maja 1994	4	III okł.
Międzynarodowe Targi Zastosowań Informatyki w Przemyśle i CAD/CAM. 7–10 czerwca 1994 r.	5	26
Konferencja Klubu Użytkowników Progressa. 15–17 czerwca 1994 r.	5	13
INFOGRYF '94. 27–29 października 1994 r.	7	10
Szkoła wobec zmieniających się metod i środków informatyki	7	25
Information Systems Concepts	7	III okł.
VIII Międzynarodowe Targi Oprogramowania SOFTARG '94. 13–15 września 1994	8	IV okł.
Oprogramowanie komputerowych systemów czasu rzeczywistego – zagadnienia związane z wytwarzaniem oprogramowania do zastosowań przemysłowych. 22–23 września 1994 r.	8	3
INFOSYSTEM '95. 10–13 kwietnia 1995 r.	11	IV okł.

Nowe książki

Akademicka Oficyna Wydawnicza	1	28
Wydawnictwa Naukowo-Techniczne	3	10
Instytut Badań Systemowych PAN	4	21
Wydawnictwa Naukowo-Techniczne	4	24
Akademicka Oficyna Wydawnicza PLJ	8	30

Wskazówki dla Autorów

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

Materiał oprócz tekstu zasadniczego powinien zawierać:

* krótki życiorys zawodowy Autora i jego zdjęcie,
* wykaz literatury,
* tabele,
* materiał ilustracyjny (rysunki, zdjęcia czarno-białe, wydruki) dołączony do artykułu (nie wklejać materiału w tekst),
* podpisy pod ilustracje.

Na osobnej stronie prosimy podać: tytuł naukowy, nazwisko i imię, nazwę zakładu pracy, numer telefonu służbowego oraz informację, jaką drogą przekazać honorarium – kasa Wydawnictwa, poczta, konto.

W związku z naliczaniem podatku dochodowego prosimy także podać:

* oprócz nazwiska i imienia – także drugie imię,
* imiona rodziców,
* datę i miejce urodzenia,
* numer identyfikacyjny PESEL,
* miejsce zameldowania,
* adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora.

Tekst artykułu prosimy dostarczyć w formie maszynopisu lub wydruku komputerowego, pisanego jednostronnie, z podwójnym odstępem (30 wierszy na stronie i 60 znaków w wierszu).

Rysunki powinny być czytelne i mogą być wykonane ołówkiem.

$1 \in N$ – długość reprezentacji osobników (liczba bitów reprezentujących chromosom),
$s: I^m \to I^m$ – operator selekcji,
$\rho: I \to R$ – operator przekształcający geny na cechy (feny) – funkcja dekodująca geny,
$\Omega \subseteq \{\omega: I \times I^m \to P \to I$ – zbiór operatorów genetycznych,
przy czym $P = \{p: I \to [0,1]\}$ oznacza zbiór rozkładów prawdopodobieństw na I,
$f: I \to R$ – funkcja przystosowania (ang. *fitness function*)
$T: I^m \to \{0, l\}$ – kryterium stopu.

W powyższym opisie przyjęto, że osobnik (genotyp) jest łańcuchem zer i jedynek o długości l. Łańcuch ten – chromosom – stanowi zestaw genów, czyli genotyp osobnika. W wielu aplikacjach jeden gen (to znaczy podłańcuch o ściśle określonej długości) jest bitowym odwzorowaniem parametrów funkcji celu (odwzorowanie to jest analogią do odwzorowania genotypu na fenotyp, fenotyp – zestaw cech osobnika – jest punktem w przestrzeni optymalizacyjnej).

W przedstawionym algorytmie populacja funkcjonuje przez sekwencyjne, w kolejnych jednostkach czasu, generowanie potomków aktualnej populacji, każda populacja funkcjonuje jedną jednostkę czasową (generację, populację). Liczebność populacji jest stała.

Etapy algorytmu genetycznego

Etap 1. Generacja populacji początkowej P^0 (wyznaczenie populacji w czasie $t = 0$).

Operacja wygenerowania populacji m osobników o losowych genotypach (z ewentualnym uwzględnieniem dopuszczalnych zakresów wartości cech).

Etap 2. Selekcja i reprodukcja osobników (wyznaczenie populacji, która będzie funkcjonować w czasie $t = t + 1$):

Etap 2.1 Wyznaczenie wartości parametrów i wartości funkcji reprezentowanych przez aktualną populację funkcjonującą w czasie t):

- Zdekodowanie genotypów wszystkich osobników na fenotypy – odpowiedniki parametrów w przestrzeni funkcji optymalizacyjnej.
- Wyznaczenie wartości funkcji celu dla każdego osobnika (wartości jego przystosowania).

Powtórzenie m razy **Etapu 2.2** i **Etapu 2.3:**

Etap 2.2 Selekcja osobników do reprodukcji:

Wylosowanie elementu, który będzie reprodukowany (prawdopodobieństwo to zależy od relatywnej oceny przystosowania osobnika). Może to być m kolejnych prób losowych z $P^t = (e_1^t, ..., e_m^t)$ zgodnie z prawdopodobieństwem:

$$p_s(e_i^t) = \frac{f(e_i^t)}{\sum_{j=1}^{m} f(e_j^t)}$$

gdzie:
$p_s(e_i^t)$ – prawdopodobieństwo wyboru do reprodukcji osobnika e_i funkcjonującego w populacji w czasie t,
$f(e_i^t)$ – wartość funkcji przystosowania osobnika e_i funkcjonującego w czasie t.

Etap 2.3. Reprodukcja elementu:

Etap 2.3.1. Operacja *crossing-over*:

- Wylosowanie osobnika, z którym reprodukowany osobnik będzie poddawany operacji *crossing-over*, prawdopodobieństwo również może zależeć od wartości przystosowawczej osobników. Losowanie może być wg p_s lub wg rozkładu równomiernego dla m elementów populacji.
- Wylosowanie punktu przerwania chromosomów: losowana jest (wg rozkładu równomiernego) liczba r z zakresu od 1 do l, gdzie l jest liczbą bitów reprezentacji chromosomu; jest to numer bitu, przed którym następuje przerwanie chromosomów i wymiana ich części.
- Utworzenie dwóch osobników – łańcuchów z wymienionymi częściami początkowych łańcuchów oraz wylosowanie jednego z nich jako osobnika po *crossing-over*. Losowanie to może być wg rozkładu równomiernego lub prawdopodobieństwo to może zależeć od przystosowania obu potomków.

Etap 2.3.2.: Operacja mutacji:

Z ustalonym jako parametr GA prawdopodobieństwem, każdy osobnik podlega losowej zmianie jego łańcucha – *mutacji*. Dla mutowanego osobnika losuje się (z jednakowym prawdopodobieństwem) numer bitu, który ma być zmieniony: jest to liczba z przedziału $1..l$. Wartość bitu o wylosowanym numerze ulega zmianie z 0 na 1 lub z 1 na 0. Otrzymany osobnik staje się elementem nowej generacji populacji, aktualnej w czasie $t + 1$.

Etap 3. Warunek stopu

Sprawdzenie, czy należy zatrzymać ewolucję. Warunkiem stopu może być osiągnięty numer generacji (czy $t = T$) lub zadana liczba generacji bez znalezienia lepszego rozwiązania i in. Jeśli warunek stopu nie zachodzi, następuje powrót do etapu 2.

Na czym polega siła algorytmów genetycznych?

Zadziwiająco prosty algorytm genetyczny jest skuteczny, ponieważ wykorzystuje obszary o wysokim współczynniku przystosowania; są to obszary docelowe. Dzieje się tak, bo każda generacja przez reprodukcję i *crossing-over* ma coraz liczniejsze potomstwo w tych właśnie obszarach. Ponieważ algorytm przy rozmnażaniu faworyzuje łańcuchy najlepiej przystosowane (pochodzące z obszarów docelowych), to będą one miały więcej potomsta w następnej generacji. Liczebność łańcuchów w danym obszarze przestrzeni poszukiwań (przestrzeni parametrów) wzrasta proporcjonalnie do statystycznej oceny wartości przystosowania w tym obszarze. Algorytm genetyczny nie potrzebuje tak dużej liczby próbek pobranych z tysięcy milionów obszarów, aby oszacować średni współczynnik przystosowania w każdym obszarze. Uzyskuje ten wynik za pomocą znacznie mniejszej liczby łańcuchów i z niewielkim, czy wręcz minimalnym wysiłkiem obliczeniowym. Aby zrozumieć takie zachowanie algorytmu, należy uświadomić sobie fakt, że pojedynczy łańcuch (element) przynależy do wszystkich obszarów definiowanych przez jakikolwiek podzbiór jego bitów. Weźmy łańcuch 00100110. Należy on jednocześnie do obszarów zdefiniowanych przez: 00****** (gdzie znak * oznacza, że w tym miejscu może być zarówno jedynka, jak i zero; nie jest to ważne, bo wartość w tym miejscu nie jest sprecyzowana), 0******0, **1**11*, itd. W literaturze przedmiotu mówi się o schematach (ang. *schemata*), w których na ustalonych miejscach są konkretne wartości (zera lub jedynki), a na pozostałych – gwiazdka oznacza nie ustaloną wartość. Duże obszary są definiowane przez łańcuchy o dużej liczbie nie sprecyzowanych bitów, na ogół więc będą próbkowane przez dużą liczbę łańcuchów (elementów) z populacji. Populacja licząca kilkaset czy kilka tysięcy łańcuchów testuje znacznie większą liczbę obszarów. Jest to najważniejsza bodajże zaleta algorytmów genetycznych w stosunku do innych metod: naturalna, wynikająca z istoty algorytmu współbieżność.

Aby w każdej generacji testować nowe obszary, jest wprowadzony do algorytmu mechanizm *crossing-over*. Istnienie tego mechanizmu utrudnia analizę efektów współbieżności, ponieważ przesuwa on łańcuch z jednego obszaru do drugiego. W efekcie tego przesuwania częstość testowania obszarów nie jest proporcjonalna do średniej wartości przystosowania w danym obszarze. Zwalnia to trochę proces ewolucji, umożliwia jednak badanie nowych obszarów, które w początkowej populacji nie wystąpiły. Prawdopodobieństwo, że potomstwo dwóch łańcuchów (elementów) opuści obszar, z którego pochodzą rodzicielskie łańcuchy, zależy od tego, jaka jest odległość między bitami definiującymi ten obszar. Potomstwo łańcucha **01**** opuści ten obszar

wtedy, gdy wylosowane miejsce *crossing-over* będzie dokładnie między trzecim a czwartym bitem łańcucha. W tym przypadku prawdopodobieństwo takiego zdarzenia byłoby 1/7 (jest siedem możliwych miejsc dla wystąpienia *crossing-over*). Skrajnym przykładem jest łańcuch: 0******1. Potomek takiego łańcucha opuści ten obszar niezależnie od miejsca, w którym wystąpi *crossing-over*.

Pojęcie *schematu* jest bardzo wygodnym narzędziem w analizie formalnej algorytmów genetycznych, umożliwiającym śledzenie efektów reprodukcji i operatorów genetycznych. Przy analizie wyróżnia się dwie cechy charakteryzujące schematy: *rząd* (ang. *order*) schematu i jego *długość* (ang. *length*). Rząd schematu jest liczbą bitów o ustalonej wartości, natomiast długość schematu definiuje się jako różnicę między numerami pierwszego i ostatniego bitu o ustalonej wartości. Przykładowo, schemat 011*1** ma rząd równy 4, a długość równą $5 - 1 = 4$, schemat *******1 ma rząd równy 1, a długość równą $8 - 8 = 0$. Analiza efektów reprodukcji i operatorów genetycznych przedstawiona jest w rozdz. 2. pracy [4]. Schematy o nieprzerwanych ciągach ustalonych bitów, definiujące określony obszar, nazywane są elementami *spójnymi* (ang. *compact building blocks*). Elementy spójne mają duże szanse przeżycia *crossing-over* i być reprezentowanymi w przyszłych generacjach proporcjonalnie do ich wartości przystosowania. Liczba takich obszarów jest znacznie większa od liczby łańcuchów (elementów) w populacji, zatem można uznać, iż algorytm genetyczny jest *inherentnie* współbieżny również z operatorem *crossing-over*.

Fundamentalne twierdzenie algorytmów genetycznych, zwane też *twierdzeniem schematu* (ang. *Schema Theorem*) wg [4] mówi, że:

$$m(H, t+1) \geq m(H,t) \cdot \frac{f(H)}{f_{sr}} \cdot \left[1 - p_c \cdot \frac{\delta(h)}{1-1} - o(H)p_m\right]$$

gdzie:
$m(H, t+1)$ – oczekiwana liczba elementów ze schematem H w czasie $t+1$,
$m(H, t)$ – liczba elementów ze schematem H w czasie t,
$f(H)$ – średnia wartość przystosowania elementów ze schematem H istniejących w czasie t,
f_{sr} – średnia wartość przystosowania wszystkich elementów w populacji w czasie t, tzn.:

$$f_{sr} = \sum_{j=1}^{n} f_j / n,$$

gdzie n – liczba elementów w populacji w czasie t,
f_j – wartość przystosowania elementu o numerze j w czasie t.
P_c – prawdopodobieństwo wystąpienia *crossing-over*
δH – długość schematu H,
l – liczba bitów w całym łańcuchu ($l - 1$ jest liczbą możliwych miejsc *crossing-over*),
$o(H)$ – rząd schematu H,
p_m – prawdopodobieństwo mutacji pojedynczego bitu w łańcuchu.

W powyższej analizie założono, że w czasie reprodukcji każdy łańcuch (element e_i) jest wybierany do reprodukcji zgodnie z prawdopodobieństwem $p_i = f_i / \sum_{j=1}^{n} f_j$, gdzie f_i – wartość przystosowania elementu o numerze $i(e_i)$.

* * *

Algorytmy genetyczne mają zastosowania w sztucznej inteligencji (AI). Na rynku AI znajdują się systemy działające na podstawie algorytmów genetycznych – dość szeroko rozpowszechniany jest system C DARWIN II firmy NovaCast (Szwecja), który jest szkieletowym systemem pozwalającym rozwiązywać różne zadania optymalizacyjne. GA są wykorzystywane jako części większych systemów eksperckich, takich jak np. stosowany w wielu firmach i krajach system ekspercki XpertRule angielskiej firmy Attar Software Ltd. Jest to generator systemów z bazą wiedzy (KBSG – *Knowledge Base System Generator*) dla konkretnych zastosowań [7]. System ten składa się z modułów umożliwiających: pozyskiwanie i reprezentację wiedzy, uczenie z dostarczonych danych (np. indukcja reguł), z modułu optymalizującego, wykorzystującego algorytmy genetyczne (łączenie systemów z bazą wiedzy z optymalizacją, projektowanie i konfiguracja aplikacji) oraz z generatora kodu źródłowego utworzonego systemu (możliwe jest uzyskanie kodu w kilku językach programowania). XpertRule znajduje zastosowanie w ponad 200 firmach japońskich m.in. firmach ubezpieczeniowych Japonii (Meiji Mutual Life Insurance Company) oraz produkujących samochody (Nissan).

Według oceny ekspertów, na rynku sztucznej inteligencji zainteresowanie algorytmami genetycznymi wynosi już ok. 8%, mimo iż według opiniodawców są one trudne do zrozumienia i brak jest narzędzi ukierunkowanych na konkretne zastosowania. Dla porównania, wg tych samych badań zainteresowanie sztuczną rzeczywistością (ang. *virtual reality*) wynosi ok. 25%, systemami eksperckimi – ok. 22%, wnioskowaniem na podstawie przykładów (*Cased-Based-Reasoning*) – ok. 20%, robotyką – ok. 13%, sieciami neuronowymi – ok. 11% [1].

GA mają również zastosowania w tworzeniu sieci neuronowych. Testuje się algorytmy genetyczne do ustalania wag połączeń, do ustalania zarówno wag jak i połączeń, a nawet do projektowania całej struktury sieci neuronowych. Powstają systemy będące kombinacją sieci neuronowych i algorytmów genetycznych. Odbywają się konferencje poświęcone algorytmom genetycznym i sieciom neuronowym, np. COGANN-92, *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, sponsorowana przez IEEE *Neural Networks Council* w kooperacji z *International Society for Genetic Algorithms*.

Omówienie zastosowań algorytmów genetycznych będzie przedmiotem następnych artykułów.

LITERATURA

[1] Coleman K.: The AI Marketplace in the Year 2000. AI EXPERT, January 1993, pp. 35–38
[2] Davis L., Steenstrup M.: Genetic Algorithms and simulated annealing: An overview. In: L. Davis (Ed.): Genetic algorithms and simulated annealing Pitman, London 1987, pp. 1–11
[3] Fogel D.B.: Evolving Artificial Intelligence. University of California, San Diego 1992
[4] Goldberg D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publ. Comp., Inc. 1989
[5] Holland J.: Adaptation in Natural and Artificial Systems. MIT Press 1992
[6] Holland J.: Algorytmy genetyczne (tłum.). W: Świat Nauki, wrzesień 1992
[7] Murphy T.: An Expert by the Rules. AI EXPERT, January 1993, pp. 35–38
[8] Riolo R.L.: Dobór naturalny w świecie bitów (tłum.). W: Świat Nauki, wrzesień 1992
[9] Schwefel H.P.: Collective Phenomena in Evolutionary Systems. Bericht Nr 233, Universität Dortmund 1987

W najbliższych numerach:

- Iwona Dąbrowska i Dorota Depowska omawiają techniki separacji i analizy kontekstowej tekstu.
- Piotr Porwik ocenia wydajność technologii „Rushmore" w bazie danych FoxPro.
- Elżbieta Kosmulska-Bochenek opisuje sposoby komunikacji między procesami i dokonuje przeglądu metod wzajemnego wykluczania się procesów w systemach rozproszonych, z uwzględnieniem ich przydatności do pracy w czasie rzeczywistym.
- Witold Staniszkis charakteryzuje projektowanie rozproszonych systemów informatycznych.

Jakość oprogramowania

Jerzy Auksztol
Katedra Informatyki Ekonomicznej
Uniwersytet Gdański

W Polsce w ostatnich latach zauważalny jest wzrost zainteresowania problematyką zapewniania jakości wytwarzanych produktów i usług. Oprogramowanie, jako jeden z ważnych elementów rynku, nie powinno być pominięte w dyskusjach nad jakością. Zanim jednak słowa *jakość oprogramowania* zostaną w różnorodnych publikacjach prasowych odmienione przez wszystkie przypadki i zanim wszystkie doniesienia prasowe o nowych produktach software'owych będą zawierały stwierdzenia *zgodne z normą ISO 9000*, warto odpowiedzieć na pytanie: co to jest jakość oprogramowania, jak definiować wymagania jakościowe dla kupowanego i tworzonego oprogramowania lub jakimi metodami i technikami można się posługiwać, aby uzyskać wysoką jakość produkowanego oprogramowania?

Zapoczątkowany niniejszym opracowaniem cykl artykułów jest poświęcony zapewnieniu jakości oprogramowania, począwszy od zdefiniowania wymagań użytkownika przez analizę, projektowanie, kodowanie, testowanie, instalację aż do jego utrzymania. Przyjęte w organizacji zasady zapewnienia jakości powinny być przestrzegane na wszystkich etapach procesu tworzenia oprogramowania.

Błędnym jest twierdzenie, że jakość oprogramowania zapewnia testowanie i można ją pominąć na innych etapach, np. w analizie, projektowaniu, czy kodowaniu. Takie podejście jest niezgodne z podstawową zasadą stosowaną w przemyśle, a potwierdzoną przez normy ISO 9000, że jakość jest gwarantowana przez system funkcjonujący na wszystkich etapach procesu tworzenia produktu lub wykonywania usługi.

W pierwszym artykule zostanie zdefiniowany zestaw mierników, umożliwiających dokładne ustalenie jakości nabywanego lub tworzonego produktu. Mierniki te zapewniają obiektywną, opartą na pomiarze liczbowym ocenę produktu programowego. W następnym artykule zostanie przedstawiony model dojrzałości organizacji, który definiuje kryteria podziału organizacji zajmujących się inżynierią oprogramowania, według wykorzystywanych w nich metod zarządzania procesem tworzenia produktów programowych. Jest to narzędzie umożliwiające ustalenie poziomu, na jakim znajduje się dana organizacja i co należy zrobić, aby uzyskać większą kontrolę nad procesem tworzenia oprogramowania. Narzędzie to zawiera również jasne kryteria umożliwiające odpowiedź na pytanie, czy konkretna organizacja, będąca wykonawcą zlecenia informatycznego i stosowane w niej metody organizacji pracy zapewnią wywiązanie się z powierzonych jej zadań. W następnych artykułach zostaną omówione szczegółowe metody i techniki zapewnienia jakości.

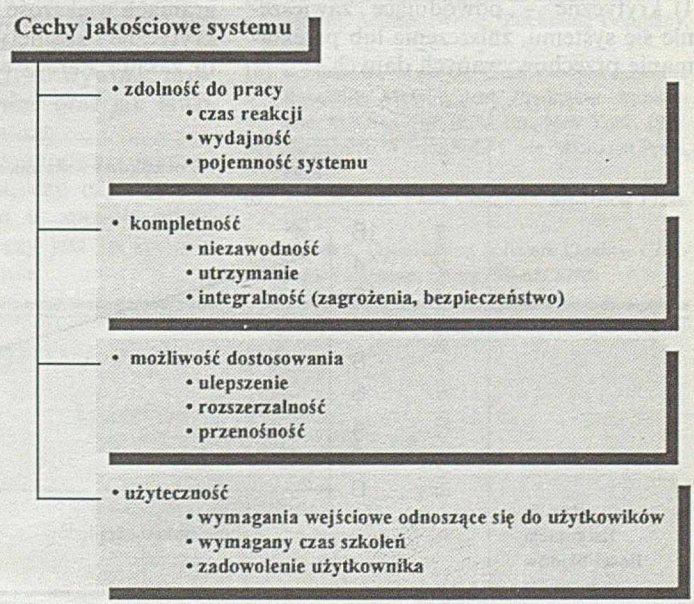
Definicja wymagań jakościowych oprogramowania aplikacyjnego

Dyskusję nad jakością oprogramowania należy rozpocząć od ustalenia mierników umożliwiających ustalenie poziomu jakości oprogramowania.

Wymagania jakościowe utożsamiane są zwykle z zestawem mierników wraz z zakresem przyjmowanych przez nie wartości. Powinny być one ustalone albo przed przystąpieniem do projektu tworzenia oprogramowania, albo w trakcie pierwszego etapu, tj. zdefiniowania wymagań użytkownika. W trakcie odbioru oprogramowania lub jego zakupu następuje porównanie początkowo zdefiniowanych wymagań z wynikami testów, na których podstawie można jednoznacznie odpowiedzieć na pytanie, czy oprogramowanie spełnia wymagania użytkownika.

Mierniki oceny wymagań jakościowych będą nazywane cechami jakościowymi, które określają „jak dobrze" system wykonuje zdefiniowane zadania. Odpowiedź na pytanie „jak dobrze?" zrealizowano przez przydzielenie cechom jakościowym wartości liczbowych z określonego zakresu. Umożliwia to jednoznaczną i obiektywną ocenę stopnia przystosowania produktu do wymagań klienta. Naturalnie, występuje silna korelacja między poziomem jakości a ceną. W przypadku ustalenia wysokich poziomów cech jakościowych powstaje konieczność dostarczenia systemu o lepszych parametrach, ale wyznaczone dla niego budżet i czas realizacji muszą być odpowiednio większe. Dlatego też konieczne staje się ustalenie optymalnego poziomu czynników ceny i jakości w zależności od dziedziny zatosowań. Inny poziom cech jakościowych powinien być wymagany dla systemu monitorowania pacjenta na oddziale intensywnej opieki medycznej, a inny dla systemu analizy płac.

Cztery grupy cech jakościowych prezentuje rys. 1.



Rys. 1. Cechy jakościowe oprogramowania

Poniżej zostanie omówiona każda z tych cech.

Zdolność do pracy

Jest to miara zdolności systemu do wykonania określonej liczby zadań. Miara ta składa się z następujących miar szczegółowych:

- Czas reakcji to pomiar reakcji systemu na pojawienie się określonego zdarzenia. Przykładem może tu być czas odpowiedzi na zapytanie o stan konta bankowego. Pomiar powinien być prezentowany w jednostkach czasu.
- Wydajność to pomiar zdolności systemu do wykonania określonych transakcji w jednostce czasu. Przykładem może być określona liczba transakcji w ciągu sekundy.
- Pojemność systemu to pojemność poszczególnych elementów systemu potrzebnych do przechowywania i przetwarzania danych objętych przez system. Wielkość ta może być ustalona na podstawie już funkcjonującego systemu lub na podstawie szacunków wymagań użytkownika.

Kompletność

Kompletność jest miarą dostosowania systemu do wykonywania zadań, jakie zostały ustalone w trakcie definiowania wymagań użytkownika. Miara ta składa się z następujących trzech miar szczegółowych:

Niezawodność. Określa stopień przystosowania systemu do wykonywania postawionych mu zadań. Cecha ta jest cechą krytyczną systemu, co oznacza, że niedotrzymanie ustalonych limitów może doprowadzić do załamania się systemu. Pomiar niezawodności systemu opiera się na błędach wykrytych w trakcie tworzenia oprogramowania, jego testowania oraz eksploatacji systemu. Błędy niezawodności można podzielić na trzy grupy:
1) krytyczne – powodujące zawieszenie się systemu, zniszczenie lub przekłamanie przechowywanych danych,
2) niekrytyczne – dostarczające niewłaściwych rezultatów, brak wyspecyfikowanych funkcji systemu,
3) kosmetyczne – nieprawidłowe rozmieszczenie formatki ekranowej, błędy w pisowni itp.

Definicja błędu niezawodności jest zagadnieniem w miarę prostym, jednak użycie tej definicji w praktyce staje się problematyczne. Podjęcie decyzji o wejściu systemu do eksploatacji jest praktycznie odpowiedzią na pytanie: ile błędów jeszcze pozostało i czy można te błędy zaakceptować. Przy odpowiedzi na to pytanie powstanie poważny problem, gdyż o ile ewidencja wykrytych błędów w systemie jest zagadnieniem czysto buchalteryjnym, to dokładne szacowanie nie wykrytych błędów jest zagadnieniem z pogranicza czarnej magii. Praktycznie żadna technika nie jest w stanie ustalić liczby pozostawionych błędów z absolutną dokładnością. W takiej sytuacji może pomóc technika oparta na systematycznym pomiarze nowo wykrytych błędów w określonym czasie. Technika ta jest oparta na stwierdzeniu, że im więcej błędów w systemie, tym łatwiej je wykryć. Innymi słowy, w określonej jednostce czasu więcej błędów zostanie wykrytych, jeżeli oprogramowanie ma większą liczbę błędów, niż w przypadku, gdy oprogramowanie jest już przetestowane i wolne od większości początkowych błędów. Zauważalna jest spadkowa tendencja wskaźnika wykrywalności błędów, tj. liczba nowo wykrytych błędów na jednostkę czasu, i przez obserwację tej tendencji można z dużą dokładnością ustalić liczbę pozostałych w systemie błędów, jak przedstawiono to na rys. 2. Teoretyczne podstawy tej techniki zawarte są w publikacji [3].

Szacunkowa wartość mówiąca o liczbie błędów pozostających w systemie jest wartościowym miernikiem jakości systemu. Liczbowo w dobrze napisanych programach większość błędów to błędy niekrytyczne i kosmetyczne. Stosowana jest tu zasada Pareto 80/20, tzn. w programach najczęściej pozostaje 80% błędów niekrytycznych i 20% błędów krytycznych. Dlatego też prawdopodobieństwo znalezienia błędu krytycznego w fazie testowania jest bardzo małe. Potwierdza to również praktyka [4]. Rozbudowanie fazy testowania nie zmniejsza liczby błędów w programach przeznaczonych do eksploatacji. Jedynie uważne i zgodne z zasadami jakości prowadzenie całego projektu umożliwia ograniczenie błędów krytycznych.

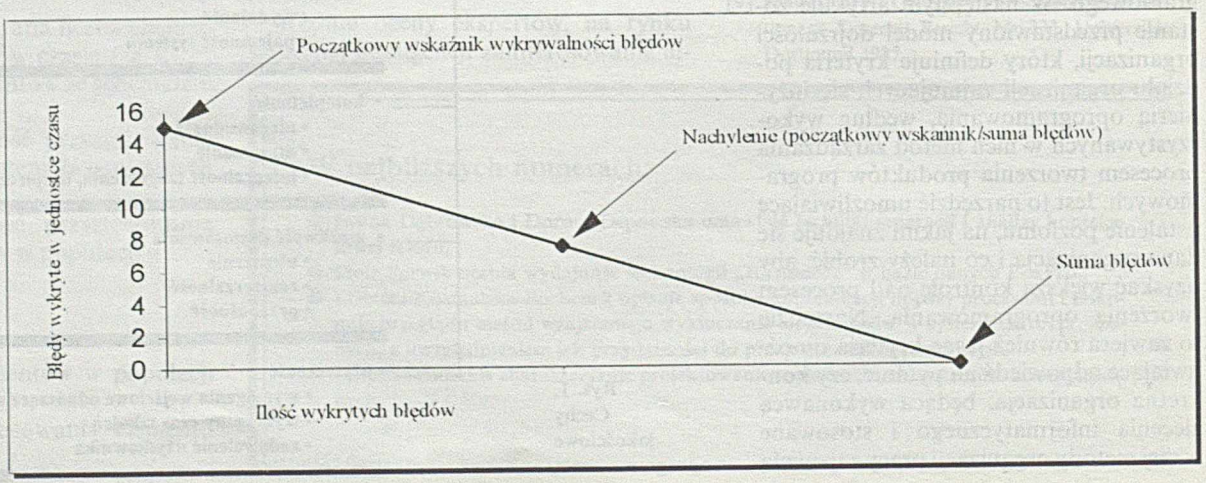
Utrzymanie. Pojęcie utrzymania odnosi się do procesu reakcji na zaistniałe błędy, a nie do procesu rozszerzenia lub ulepszenia systemu. Głównymi miarami określającymi tę cechę jest czas reakcji na raportowane błędy, jak również czas potrzebny do przywrócenia systemu do pracy.

Integralność. Cecha integralności odnosi się do zagrożeń i odpowiadających im mechanizmów bezpieczeństwa systemu. Integralność obejmuje zagrożenia powodowane przez ludzi (zamierzone lub nie) lub przez maszyny (oprogramowanie i sprzęt). Integralność można zmierzyć przez szacowanie częstotliwości wystąpienia poszczególnych zagrożeń i siły zabezpieczeń przeciwko nim. Zagrożenia należy rozpatrywać według ich typu, ogólny zaś wzór na wartość cechy integralności można przedstawić następująco:

$$Integralność = \sum[1 - zagrożenie * (1 - bezpieczeństwo)]$$

Przystosowanie

Przystosowanie jest miarą możliwości systemu do przeprowadzenia zmiany. Teoretycznie rzecz ujmując, każdy system można przystosować do pożądanego stanu, jednak zaangażowanie zasobów w postaci pieniędzy, ludzi i czasu może być diametralnie różne. Dlatego też przystosowanie określa możliwość systemu do przeprowadzenia zmiany, ale w sposób efektywny. Przystosowanie obejmuje następujące trzy miary szczegółowe:



Rys. 2. Estymacja ilości błędów w systemie

- **Ulepszenie.** Jest to cecha mówiąca o możliwościach dokonania drobnych zmian i ulepszeń w systemie. Podstawą dokonania zmian nie są powstałe błędy, ale chęć ulepszenia systemu. Podstawową miarą jest czas potrzebny do zmiany stanu systemu opierając się na jasnej definicji obszaru zmiany, np. jedna instrukcja lub 2000 rekordów lub 10 stron instrukcji obsługi.
- **Rozszerzalność.** Jest to cecha mówiąca o kosztach związanych z rozbudową systemu o dodatkowe elementy, które w znaczny sposób ingerują w strukturę logiczną funkcjonującego systemu. Zmiany dokonane w systemie generują skutki uboczne, takie jak: zmniejszenie integralności, zmniejszenie efektywności itp. Wszystkie te skutki uboczne powinny być ponownie przetestowane, co wiąże się z dodatkowymi kosztami. W przypadku stosowania narzędzi wysokiego poziomu (systemu zarządzania relacyjnymi bazami danych, języki czwartej generacji), poziom kosztów związanych z rozszerzeniem systemu jest stosunkowo niski, ponieważ uboczne skutki wprowadzania zmian są niewielkie, natomiast stosowanie narzędzi niskiego poziomu (języki trzeciej generacji) wiąże się z większymi kosztami testowania dokonanych zmian. Przy podejmowaniu decyzji o budowie systemu należy ustalić rodzaj stosowanych narzędzi. Dla narzędzi wysokiego poziomu początkowy koszt stworzenia systemu jest stosunkowo wysoki, co wiąże się z zakupem drogich narzędzi programistycznych, natomiast koszt rozszerzenia systemu jest niski. Dla narzędzi niskiego poziomu koszt stworzenia oprogramowania jest niski, ale koszty utrzymania, ulepszania, rozszerzania i przenoszenia na inne platformy jest bardzo wysoki.
- **Przenośność.** Jest to cecha mówiąca o kosztach przeniesienia funkcjonującego systemu na inną platformę sprzętowo-programową, np. koszt przeniesienia systemu ze środowiska sieci NOVELL na środowisko UNIX. Przenośność wycenia się w procentach w stosunku do kosztów poniesionych na stworzenie pierwotnego systemu. Dla narzędzi wysokiego poziomu (języki czwartej generacji), koszt przeniesienia systemu waha się w granicach od 10 do 15%, natomiast dla narzędzi niskiego poziomu – od 50 do 90%.

Użyteczność

Użyteczność jest miarą przystosowania użytkowników do praktycznego wykorzystywania systemu. Miara ta w ostatnich latach nabiera znaczenia, ponieważ na rynku oprogramowania panuje duża konkurencja, a liczba dostępnego oprogramowania jest bardzo duża. Coraz mniej jest firm, które mają własną wyspecjalizowaną komórkę informatyczną, odpowiedzialną za tworzenie systemów informatycznych dla całego przedsiębiorstwa. Coraz częściej wybiera się standardowe oprogramowanie, które następnie dopasowuje się do potrzeb przedsiębiorstwa. Czas użytkownika jest na tyle cenny, że chce on użytkować oprogramowanie bez żadnej frustracji. Dlatego też cecha użyteczności dla twórców oprogramowania jest równoznaczna z decyzją „być albo nie być". Okres, w którym można stracić klienta po pierwszym zetknięciu się z oprogramowaniem, jest bardzo krótki i co najważniejsze – staje się coraz krótszy.

Użyteczność systemu najczęściej jest wyrażona w liczbie godzin potrzebnych do przeszkolenia osoby o ustalonych kwalifikacjach, jak również przez pomiar czasu potrzebnego do zrealizowania określonego zadania w systemie. Im bardziej oprogramowanie odpowiada ustalonym w przemyśle standardom, tym mniej czasu potrzeba na przeszkolenie, a cecha użyteczności osiąga lepszy poziom.

W ostatnich latach w wielu firmach produkujących oprogramowanie powstały laboratoria użyteczności, które towarzyszą rozwojowi oprogramowania na każdym jego etapie. Po każdym teście twórcy uzyskują wiele informacji na temat produktu. Dotyczą one:

- dostępności – jak łatwo wywołać program i nim operować oraz jak ławo zakończyć z nim pracę;
- adekwatności – czy zapewnia wymagania użytkownika, czy użytkownik obsługuje program w sposób przez niego pożądany i czy jest to sposób łatwy do opanowania;
- efektywności – czy użytkownik wykonuje to, co chce, poświęcając minimum czasu;
- kompletności – czy użytkownik rozumie strukturę oprogramowania, czy występuje pomoc systemowa, czy jest dołączona prosta w użyciu i napisana zgodnie ze standardami dokumentacja.

Użyteczność stawia na pierwszym miejscu klienta i jego odczucia związane z użytkowaniem oprogramowania. Cecha ta powinna być brana pod uwagę od początku projektowania systemu, natomiast wygłaszanie stwierdzeń typu: *to tylko kosmetyka* lub *można to zrobić później*, jest początkowym objawem niepowodzenia projektu oprogramowania.

* * *

Prezentowane cechy jakościowe mają zadanie umożliwić skonstruowanie takiego zestawu kryteriów oceny nabywanego lub tworzonego oprogramowania, który byłby zróżnicowany w zależności od organizacji, a także branży zastosowań. Inne kryteria będą stosowane w banku, inne w służbie zdrowia, a jeszcze inne w przedsiębiorstwie produkcyjno-usługowym. Przy podejmowaniu decyzji o zakupie lub tworzeniu oprogramowania istotne jest stosowanie obiektywnych mierników liczbowych, umożliwiających jednoznaczną ocenę, czy produkt odpowiada wymaganiom danej organizacji.

W następnym artykule zostanie przedstawiona metoda oceny funkcjonowania organizacji zajmującej się inżynierią oprogramowania, nazywana modelem dojrzałości organizacji.

LITERATURA

[1] Abelow D.: Usability testing tutorial. Computer Language, 03, pp. 41–47, 1993

[2] Gilb T.: Principles of software engineering management. Addison-Wesley Publishing Company, 1988

[3] Iannino A., Musa J., Okumoto K.: Software Reliability Measurement, Prediction, Application. McGraw-Hill Book Co., New York, 1987

[4] Maguire S.: Writing Solid Code. Microsoft Press, 1993

[5] McConnel S.: Code Complete. Microsoft Press, 1993

[6] Walsh J.: Determining Software Quality. Computer Language, 04, pp. 57–65, 1993.

Ogłoszenia prosimy zgłaszać:

pisemnie
Red. INFORMATYKI
Pl. Inwalidów 10, p. 104, 01-552 Warszawa

Dział Reklamy i Marketingu
ul. Mazowiecka 12, 00-950 Warszawa
skr. poczt. 1004

telefonicznie
nr 39-14-34 lub 27-43-66

telefaksem
nr 39-14-34
26-80-16
z zaznaczeniem – Red. INFORMATYKI

AWEK akustyczne wejście do komputera

Mimo niewątpliwych ulepszeń w oprogramowaniu ułatwiającym komunikację człowieka z komputerem, jest oczywiste, że wąskim gardłem w komunikowaniu się człowieka z komputerem jest sposób wprowadzania i wyprowadzania informacji. Klawiatura i monitor, mysz, bo są to typowe i najbardziej rozpowszechnione środki komunikowania, wykorzystują zmysł wzroku i własności ruchowe ręki operatora. Zupełnie nie wykorzystuje się mowy i zmysłu słuchu. Z codziennej praktyki wiadomo, że mowa jest najefektywniejszym i powszechnym sposobem komunikowania się ludzi. Właśnie mowa powinna też stać się jednym z mediów komunikacji człowieka z komputerem. Rozpoznawanie elementów lingwistycznych przez człowieka opiera się na analizie docierającego do niego akustycznego sygnału mowy, ale również na znajomości istniejących w danym języku kontekstowych, gramatycznych i semantycznych ograniczeń. Realizacja akustycznego wejścia do komputera (AWEK) wymaga przede wszystkim rozwiązania problemu automatycznego rozpoznawania mowy (ARM).

Czesław Basztura
Instytut Telekomunikacji i Akustyki
Politechnika Wrocławska

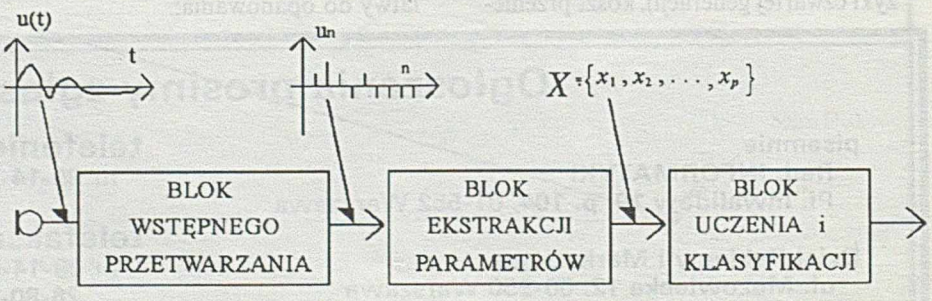
Automatyczne rozpoznawanie mowy należy do wspólnej grupy badań objętych nazwą „analiza i przetwarzanie mowy". Badania te obejmują również **automatyczne rozpoznawanie głosów (ARG), syntezę mowy (SM), techniczną analizę mowy** oraz **akustyczną diagnostykę narządu głosu** (część akustycznej diagnostyki medycznej). Komputery i systemy wykorzystujące przekazywane im informacje za pośrednictwem sygnału mowy mają następujące zalety:

- Sterowanie i przekazywanie danych głosem może być realizowane znacznie szybciej niż za pomocą klawiatury alfanumerycznej. Szybkość komunikowania się operatora sprzętu cyfrowego za pośrednictwem klawiatury nie przekracza na ogół 0,5 słowa/s, a spontaniczna mowa przebiega ze średnią szybkością od 2,5 do 3,6 słów/s.
- Przekazywanie informacji do maszyny za pomocą sygnału mowy umożliwia zwolnienie rąk operatora, które mogą być równocześnie wykorzystane do manipulowania innymi układami sterowania.
- Przekazywanie informacji głosem może odbywać się w różnych nietypowych sytuacjach i położeniach operatora (np. pod wodą czy w stanie nieważkości w kabinie pojazdu kosmicznego).
- Czas reakcji głosowej jest znacznie krótszy niż rekcji ruchowej, co pozwala na wykorzystanie sterowania głosem tam, gdzie jest wymagane szybkie sterowanie lub błyskawiczne wyłączenie układu, urządzenia czy maszyny na skutek zaistniałego niebezpieczeństwa.
- Układy sterowania głosem pozwalają w znacznym stopniu złagodzić skutki ludzkiego kalectwa, objawiającego się ograniczeniem zdolności ruchowych rąk lub nóg.
- Sterowanie głosem nie wymaga od operatora specjalnego przygotowania czy treningu.

Automatyczne rozpoznawanie mowy ARM jest realizowane w procesie sekwencji procedur, które w dużym uproszczeniu można zgrupować w trzy podstawowe bloki (rys.).

W bloku wstępnego przetwarzania następuje odbiór sygnału mowy z mikrofonu jako źródła wtórnego sygnału oraz takie jego wstępne przetworzenie, aby mógł być on przydatny do dalszej obróbki cyfrowej w następnym bloku ekstrakcji parametrów. Wstępne przetworzenie obejmuje zazwyczaj takie wzmocnienie sygnału, aby jego poziom dopasować do zakresu przetwornika analogowo-cyfrowego a/c, oraz filtrację sygnału, czyli ograniczenie pasma częstotliwości od dołu i od góry, tak aby górna częstotliwość pasma sygnału mowy była co najmniej dwukrotnie niższa od częstotliwości pobierania próbek przez przetwornik a/c.

W bloku ekstrakcji parametrów następuje obróbka i analiza sygnału mowy w celu otrzymania wartości parametrów niosących informację o treści wypowiedzi, ale możliwie niezależnych od indywidualnych cech głosu mówiącego. Parametry te, to wartości liczbowe cech fizycznych sygnału, które muszą mieć właściwości dystynktywne, a więc pozwalające na rozróżnienie treści wypowiedzi. Aby otrzymać takie parametry, uruchamiane są określone **procedury analizy** i **ekstrakcji,** które po zmierzeniu interesujących wartości parametrów, odpowiednio je porządkują do postaci wektora x lub macierzy X. Te uporządkowane zbiory parametrów są nazywane obrazami całej wypowiedzi lub jej odcinka, czyli segmentu wypowiedzi. Obrazy wypowiedzi są przesyłane do bloku **klasyfikacji**, gdzie odpowiednie procedury, opierając się na określonym algorytmie dokonują porównania nadchodzących ciągów obrazów wypowiedzi ze znajdującymi się w pamięci **wzorcami**, czyli obrazami zawierającymi uogólnione wartości i opisy danych rozpoznawanych klas dźwięków. Klasą mogą być **fonemy,** alofony, sylaby, wyrazy lub nawet całe zdania. Duże nadzieje, co wynika z ostatnich naszych badań oraz docierających doniesień literaturowych, pokłada się w przyjęciu jako podstawowych jednostek językowych, **diafonów**, czyli połączeń międzyfonemowych. Obrazy wzorcowe oraz inne dodatkowe dane, np. językowe, ułatwiające proces rozpoznawania, są tworzone wcześniej, w trakcie procesu zwanego **uczeniem.** Wyniki klasyfikacji wprowadzane są



Podstawowe bloki automatycznego rozpoznawania mowy

w postaci kodu lub ciągów kodowych i mogą stać się danymi wejściowymi systemu dialogowego SD.

Oprócz potrzeby rozpoznawania treści, w niektórych sytuacjach istnieje potrzeba rozpoznawania głosu operatora lub rozmówcy. W sygnale mowy, oprócz informacji określających treść wypowiedzi, zawarte są również informacje związane z wewnętrzną strukturą źródła wiadomości, a wśród nich **informacje osobnicze,** charakteryzujące indywidualne cechy głosu mówcy. Te informacje wynikają z różnej budowy organów artykulacyjnych u różnych osób, z różnych nawyków artykulacyjnych nabytych w trakcie uczenia się mówienia lub różnego stopnia opanowania języka. Cechy osobnicze głosu zawarte w sygnale mowy umożliwiają rozpoznanie mówcy na podstawie jego wypowiedzi. Z rozpoznawaniem głosów, a tym samym mówców, spotykamy się w licznych sytuacjach życia codziennego, np. w czasie rozmowy telefonicznej rozpoznajemy głos znajomej nam osoby, a w czasie słuchania audycji radiowej poznajemy głos znanego nam aktora.

Przedstawione sytuacje rozpoznawania osób na podstawie głosu nazywa się słuchowym, subiektywnym rozpoznawaniem głosów. Rozpoznawanie głosów z udziałem człowieka odbywa się jeszcze przy wykorzystywaniu metody wizualnej na podstawie spektrogramów, otrzymanych zautomatyzowaną analizą spektrograficzną (widmowo-czasową) określonej frazy, jednakowej dla wszystkich badanych głosów. Obecnie metoda ta jest mocno kwestionowana. Współczesny rozwój społeczno-gospodarczy spowodował pojawienie się nowego rodzaju zapotrzebowania na rozpoznawanie osób na podstawie ich wypowiedzi. Wiele transakcji handlowych i nie tylko handlowych mogłoby być zawieranych przez telefon, gdyby istniała możliwość natychmiastowego rozpoznania mówcy w sposób automatyczny, za pomocą w pełni obiektywnych kryteriów.

Duże nadzieje wiąże się z zastosowaniem automatycznego rozpoznawania osób na podstawie ich wypowiedzi w kryminalistyce, ochronie dostępu do zastrzeżonych zbiorów informacji przechowywanych w pamięciach systemów komputerowych lub dla sterowania głosem urządzeń przez uprawnione osoby. Automatyczne rozpoznawanie głosów mówców ARG polega na realizacji reguł decyzyjnych na mierzalnych, w sposób obiektywny, cechach sygnału mowy w celu określenia, czy dana wypowiedź należy do określonego mówcy lub zbioru mówców.

W metodach automatycznych udział człowieka w eksperymentach rozpoznawania jest ograniczony do minimum, przy czym jest on nieco mniejszy w półautomatycznych metodach interakcynych, niż w metodach w pełni zautomatyzowanych. Metody automatyczne wydają się najbardziej obiecujące oraz rokujące największą nadzieję na praktyczną implementację i do nich zostanie ograniczone dalsze omawianie problemów związanych z rozpoznawaniem głosów.

dokończenie na s. 26

Projekt AWEK

Zasadniczymi problemami, których efektywne rozwiązanie pozwoli pomyślnie zrealizować projekt AWEK, były:

- Badania, pozwalające ustalić zasady tworzenia słowników i utworzenie słowników modułowych.
- Opracowanie skutecznych metod ekstrakcji parametrów przenoszących informacje lingwistyczne i osobnicze.
- Aplikacja algorytmów uczenia i rozpoznawania, uwzględniających rzeczywiste uwarunkowania działania AWEK.
- Uzyskanie danych na poziomie akustycznym, pozwalających na opracowanie w przyszłości systemów ARMC.

W ramach realizowanego projektu od września 1991 r. do maja 1994 r., a więc w okresie obejmującym ok. 90% całkowitego czasu realizacji projektu, wykonano następujące zadania badawcze:

W ramach Zadania 1, obejmującego IV kwartał 1991 r., wykonano następujące prace:

- Opracowano szczegółową koncepcję prowadzenia badań z uwzględnieniem zamierzonych celów, ludzkiego potencjału badawczego, możliwości aparaturowych oraz otrzymywanych środków finansowych.
- Dokonano szerokiego przeglądu literaturowego, odnoszącego się do przewidywanych badań projektu.
- Opracowano zasady doboru słowników o strukturze modułowej. Ustalono cztery podstawowe moduły słownika systemu badawczego AWEK.
- Przeanalizowano potrzeby oraz dokonano kompletacji aparatury.

W ramach następnych zadań, obejmujących lata 1992, 1993 i początek 1994 r., wykonano następujące prace:

- Przeprowadzono bieżące studia dotyczące ARM i ARG oraz wykonano opracowanie dotyczące podstaw teoretycznych ARM i ARG.
- Opracowano procedurę i program komputerowy automatycznej transkrypcji z kodu ortograficznego na fonetyczny.
- Zaprojektowano i wykonano układy analogowe wejść akustycznych obejmujące wzmacniacze, filtry antaliasowe i układy dopasowujące do kart przetworników analogowo-cyfrowych.
- Wybrano oraz opracowano algorytmy przetwarzania i ekstrakcji parametrów przenoszących informacje lingwistyczne i osobnicze.
- Wykonano (w języku TP 7.0) programy ekstrakcji czterech zespołów parametrów według ww. algorytmów.
- Opracowano i wygłoszono referaty na:
  - Konferencji Komitetu Akustyki PAN w Mądralinie, luty 1993 r.,
  - XXXIX Otwartym Seminarium z Akustyki OSA'92, Kraków 22–24 września 1992 r.,
  - OSA '93, Rzeszów-Polańczyk, 14–17 września 1993,
  - seminarium Polskiego Towarzystwa Fonetycznego, 10–11 grudnia 1993 r.
- Opracowano cztery tomy (piąty w opracowaniu) raportu końcowego.
- Wydano książkę pt. „Rozmawiać z komputerem".
- Opracowano procedurę i program (TP 6.0) automatycznego wyszukiwania początku i końca wyrazu.
- Dokonano wstępnych nagrań słowników modułowych i słów kluczowych.
- Przetworzono na postać cyfrową słowniki i słowa kluczowe.
- Opracowano system badawczo-testowy AWEK (komputer IBM PC486, oprogramowanie TP 7.0), umożliwiający wszechstronne badania i testy różnych wersji systemów ARM.
- Zaimplementowano (na komputerze przenośnym) demonstracyjną wersję AWEK oraz przeprowadzono wstępne testy obydwu systemów.
- Zainicjowano opracowanie wersji AWEK uwzględniającej wyspecjalizowane układy DSP typu TMS320C25.
- Zainicjowano badania mające na celu stworzenie podstaw ARMC, opierając się na wykorzystaniu diafonów i sieci ANN.

SYSTEMY CZASU RZECZYWISTEGO

REDAGUJE ZESPÓŁ: dr inż. Zbigniew FRYŹLEWICZ (Politechnika Wrocławska), prof. dr hab. Janusz GÓRSKI (Francusko-Polska Wyższa Szkoła Nowych Technik Informatyczno-Komunikacyjnych w Poznaniu), prof. dr hab. inż. Zbigniew HUZAR (Politechnika Wrocławska), dr inż. Krzysztof SACHA (Politechnika Warszawska), prof. dr hab. inż. Tomasz SZMUC (Akademia Górniczo-Hutnicza), dr inż. Zbigniew ZIELIŃSKI (Wojskowa Akademia Techniczna), mgr inż. Zdzisław ŻURAKOWSKI (Fabryka Wagonów PAFAWAG we Wrocławiu) – przewodniczący zespołu.

ROK I ● Grudzień ● 1994 ● 3

System operacyjny QNX jako narzędzie tworzenia aplikacji czasu rzeczywistego

JĘDRZEJ UŁASIEWICZ
Instytut Cybernetyki Technicznej
Politechnika Wrocławska

ZBIGNIEW FRYŹLEWICZ
Centrum Informatyczne
Politechnika Wrocławska

Systemy czasu rzeczywistego są zaliczane do najbardziej złożonych i kosztownych produktów programowych. Stąd dobór właściwych narzędzi do ich budowy jest istotnym zagadnieniem. Wymagania narzędziowe do budowy aplikacji czasu rzeczywistego zostały wyspecyfikowane w pracy [10]. Dobór narzędzi zależy od wielu czynników, a w tym specyfiki problemu i jego skali, przewidywanego okresu eksploatacji, wymaganego stopnia niezawodności systemu, kwalifikacji zespołu implementującego oraz personelu obsługującego, dostępności serwisu platformy programowej itp. Błąd popełniony na etapie doboru narzędzi może mieć długotrwałe i kosztowne konsekwencje.

Do najistotniejszych decyzji, jakie muszą być podjęte we wstępnej fazie projektu zalicza się wybór systemu operacyjnego. Pomocna jest tu dostępność norm POSIX 1003.1 na interfejs między aplikacją a systemem operacyjnym oraz POSIX 1003.4 na rozszerzenia czasu rzeczywistego. Użycie spełniającego standard systemu operacyjnego zapewnia:

- możliwość przenoszenia kodu źródłowego aplikacji na inne platformy programowe,
- niezależność od platformy sprzętowej,
- łatwiejszy dostęp do personelu znającego system.

System QNX 4 spełnia powyższe normy POSIX. W artykule zostały opisane istotniejsze, z punktu widzenia projektanta systemu czasu rzeczywistego, własności systemu operacyjnego QNX 4. W zakończeniu dokonano porównania systemu QNX z dwoma innymi systemami spełniającymi normy POSIX: LynxOS oraz VENIX.

* Artykuł jest oparty na referacie wygłoszonym na ogólnopolskiej konferencji naukowo-technicznej pt.: „Oprogramowanie komputerowych systemów czasu rzeczywistego – zagadnienia związane z wytwarzaniem oprogramowania do zastosowań przemysłowych", która odbyła się we Wrocławiu w dniach 22–23.09.1994 r.

Ogólna charakterystyka systemu QNX

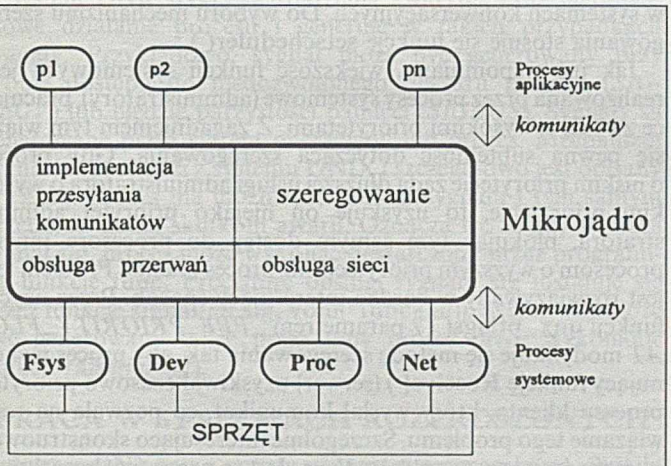
QNX jest wielozadaniowym, sieciowym systemem operacyjnym czasu rzeczywistego. Jego pierwsza komercyjna wersja pojawiła się w 1982 r. i oparta była na pracach D. Dodge'a i G. Bella z University of Waterloo w Ontario [4]. System był od podstaw konstruowany dla komputerów PC i miał, mimo skromnych w tamtym czasie możliwości komputerów klasy PC, zapewniać wielozadaniowość, wielodostęp, pracę w trybie ochronnym mikroprocesora, czas rzeczywisty, ochronę plików oraz integrację zasobów stacji roboczych przez sieć lokalną. Cele te zostały osiągnięte głównie dzięki oryginalnej architekturze systemu opartej na przekazywaniu komunikatów między zadaniami systemu. QNX zapewniał możliwości porównywalne z systemem UNIX przy znacznie skromniejszym zapotrzebowaniu na zasoby. Pierwotna wersja systemu została rozwinięta do wersji 2.21, która ma ponad 200 tys. instalacji i jest rozwijana do chwili obecnej. Dokładniejsze informacje o tych wersjach systemu są zawarte w publikacji [5]. W 1990 r. powstała wersja 4.0, która zapoczątkowała nową linię rozwojową systemu, zgodną z wymaganiami normy POSIX. Obecnie ukazała się 32-bitowa wersja systemu QNX4.2, która zapewnia wiele unikalnych możliwości. Jest to wersja, która może być uznana za w pełni dojrzały system operacyjny czasu rzeczywistego [2, 5, 6]. W stosunku do swoich możliwości system ma wyjątkowo skromne zapotrzebowanie na zasoby. Wystarcza procesor 286, 4MB pamięci RAM i ok. 10 MB przestrzeni dysku. Głównymi domenami zastosowań QNX-a są systemy krytyczne ze względu na bezpieczeństwo, a w tym systemy ochrony życia, sterowania obiektami, transakcyjne i telekomunikacyjne. Producentem systemu jest firma QNX Software Systems Ltd. z siedzibą w Kanadzie.

Architektura systemu

QNX (rys. 1) składa się z mikrojądra (ang. *microkernel*) oraz zbioru usługowych procesów systemowych nazywanych ad-

ministratorami. Komunikacja między procesami aplikacyjnymi a systemowymi oraz wzajemnie między procesami systemowymi odbywa się za pomocą sychronicznego przesyłania komunikatów (ang. *messages*). Mikrojądro zajmuje tylko 8 KB pamięci i realizuje następujące funkcje:

- przesyłanie komunikatów,
- szeregowanie procesów,
- obsługa przerwań niskiego poziomu,
- podstawowy interfejs z siecią lokalną.



Rys. 1. Uproszczona struktura systemu QNX

System zawiera zbiór podstawowych administratorów realizujących funkcje leżące w ich zakresie.
Podstawowi administratorzy, to:

Proc – administrator procesów (zarządzanie pamięcią, tworzenie i kasowanie procesów),
Dev – administrator urządzeń (obsługa urządzeń we-wy),
Fsys – administrator plików (obsługa plików),
Net – administrator sieci (obsługa sieci).

System QNX nie ma monolitycznego jądra, jak to występuje w innych systemach. Granica między procesami aplikacyjnymi a systemowymi nie jest wyraźnie zakreślona i użytkownik może samodzielnie dodawać własnych administratorów. Dokładniejszy opis struktury systemu QNX jest zawarty w jego dokumentacji [2, 6].

Komunikacja międzyzadaniowa

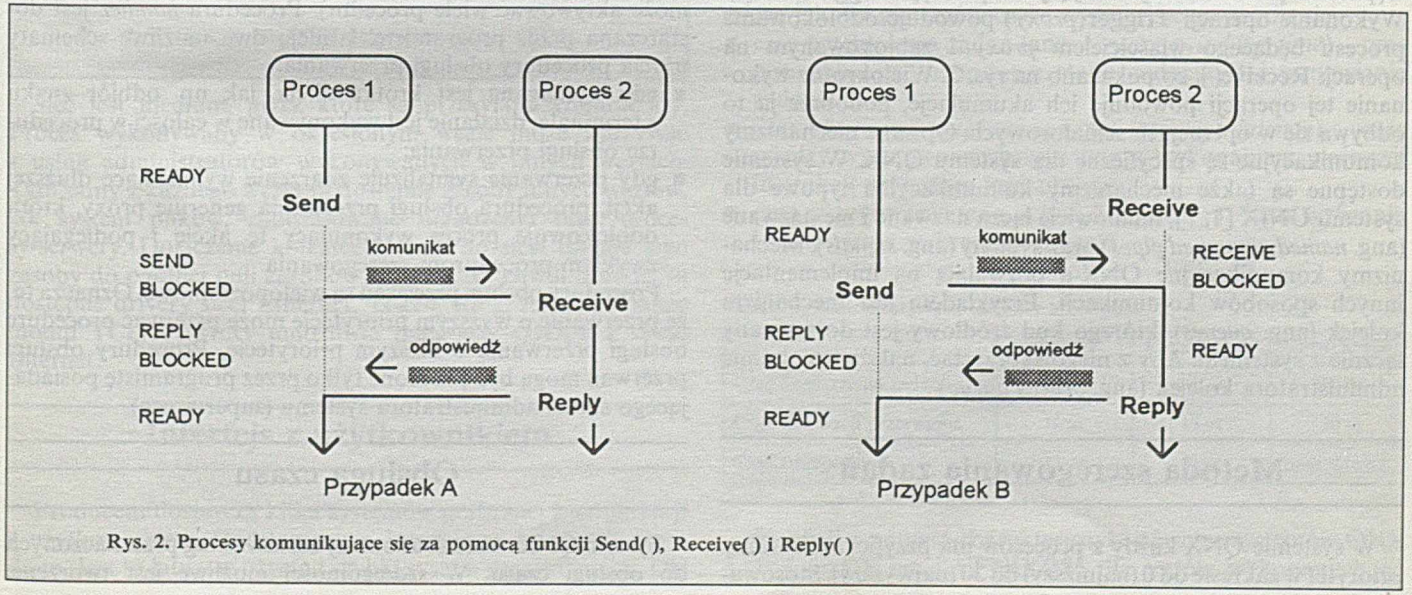
System QNX zawiera wygodne dla zastosowań czasu rzeczywistego mechanizmy komunikacyjne. Są to komunikaty synchroniczne i proxy. Przesyłanie komunikatów leży u podstaw architektury systemu QNX. Komunikat jest pakietem bajtów przesyłanym między procesami. Do obsługi przesyłania komunikatów służą trzy podstawowe funkcje: **Send( )**, **Receive( )** i **Reply( )**.

Funkcja **Send**(*pid, smsg, rmsg, s_bytes, r_bytes*) służy do wysłania komunikatu *smsg* długości *s_bytes* do procesu o identyfikatorze *pid*. Proces przesyła odpowiedź zwrotną umieszczając ją w buforze *rmsg*. Parametr *r_bytes* jest ograniczeniem dotyczącym długości tej odpowiedzi. Po wykonaniu funkcji **Send( )** proces bieżący jest zawieszany aż do czasu otrzymania odpowiedzi zwrotnej. Przechodzi wtedy kolejno od stanu *READY* do stanów *SEND_BLOCKED* i *REPLY_BLOCKED*, a po otrzymaniu odpowiedzi – z powrotem do stanu *READY*.

Funkcja **Receive**(*pid, rmsg, r_bytes*) odbiera komunikat od procesu *pid* (lub od dowolnego procesu, gdy *pid* = 0) i kopiuje jego zawartość do bufora *rmsg* organiczonego do długości *r_bytes*. Funkcja zwraca identyfikator procesu nadającego komunikat. Proces przechodzi do stanu *RECEIVE_BLOCKED* wtedy, gdy wykonuje funkcję **Receive( )** i brak jest skierowanych do niego komunikatów.

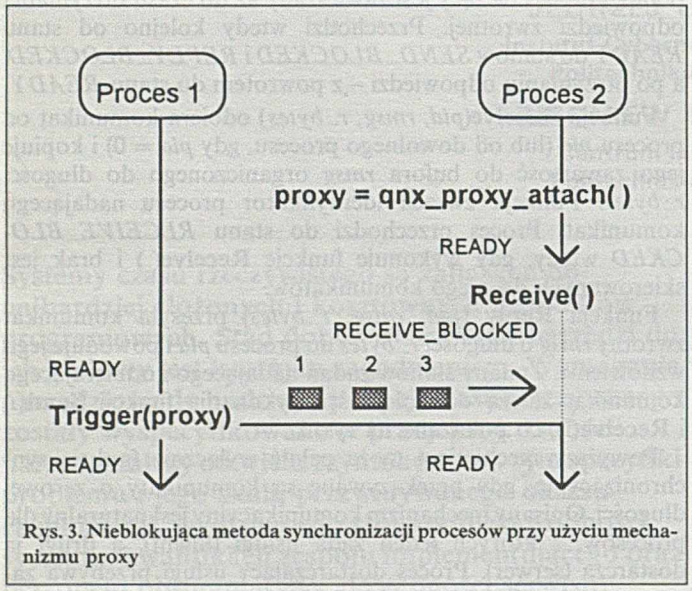
Funkcja **Reply** (*pid, rmsg, r_bytes*) przesyła komunikat zwrotny *rmsg* o długości *r_bytes* do procesu *pid* i powoduje jego wznowienie. Zmiany stanów zadań nadającego i odbierającego komunikat zależą od kolejności wykonania funkcji **Send( )** i **Receive( )**, co pokazano na rys. 2.

Powyższy mechanizm może pełnić wyłącznie funkcje synchronizacyjne, gdy przekazywane są komunikaty o zerowej długości. Opisany mechanizm komunikacyjny jest naturalny dla procesów, z których jeden żąda usługi (klient), a drugi ją dostarcza (serwer). Proces dostarczający usługi przebywa zazwyczaj w stanie *RECEIVE_BLOCKED* w oczekiwaniu na zgłoszenie. Proces żądający usługi zgłasza żądanie wykonując funkcję **Send( )** i przechodzi w stan zablokowania. Rodzaj żądanej usługi jest wyspecyfikowany w parametrze *smsg*. Wykonanie funkcji **Send( )** powoduje odblokowanie serwera. Przesyła on komunikat zwrotny, powodując odblokowanie klienta i przechodzi ponownie w stan zablokowania lub przystępuje do



Rys. 2. Procesy komunikujące się za pomocą funkcji Send( ), Receive( ) i Reply( )

obsługi kolejnego zgłoszenia. Oczywiście może się zdarzyć, że do procesu dostarczającego usługę zgłosi się jednocześnie więcej procesów niż jest on w stanie obsłużyć. W takim przypadku tworzona jest, uporządkowana według priorytetów, kolejka zadań wykonujących operację **Send( )**. Wywołanie funkcji **Send( )** i **Receive( )** potencjalnie blokuje proces. System zawiera mechanizm timerów zapewniający ograniczenie maksymalnego czasu zablokowania procesu, co opisano poniżej w punkcie „Obsługa czasu". W systemie QNX komunikaty mogą być transmitowane przez dowolne medium (pamięć operacyjną, sieć lokalną, złącze szeregowe). Stąd nie ma szczególnego znaczenia, gdzie znajduje się proces serwera. Może on być w dowolnym węźle systemu. Jeżeli procesy klienta i serwera przebiegają w różnych węzłach, to czas transmisji komunikatów zależy od przepustowości łącza.



Rys. 3. Nieblokująca metoda synchronizacji procesów przy użyciu mechanizmu proxy

System dostarcza również nieblokującego mechanizmu synchronizacyjnego w postaci tzw. *proxy*. Brak jest zadowalającego tłumaczenia tego terminu – mechanizm ten przypomina najbardziej sygnały systemu UNIX. Proces, który ma być odblokowany przez *proxy*, musi ten obiekt utworzyć wywołując funkcję *proxy* = **qnx_proxy_attach( )**. Następnie *proxy* musi być udostępniona procesom wykonującym operację **Trigger**(*proxy*). Wykonanie operacji **Trigger**(*proxy*) powoduje odblokowanie procesu będącego właścicielem *proxy* i zablokowanym na operacji **Receive( )**, co pokazano na rys. 3. Wielokrotne wykonanie tej operacji powoduje ich akumulacje, podobnie ja to odbywa się w operacjach semaforowych. Opisane mechanizmy komunikacyjne są specyficzne dla systemu QNX. W systemie dostępne są także mechanizmy komunikacyjne typowe dla systemu UNIX [1, 7], mianowicie łącza nazwane i nie nazwane (ang. *named, unnamed pipes*) oraz sygnały (ang. *signals*). Mechanizmy komunikacyjne QNX-a pozwalają na implementację innych sposobów komunikacji. Przykładem jest mechanizm kolejek (ang. *queues*), którego kod źródłowy jest dostarczany łącznie z systemem. Aby z niego skorzystać, należy uruchomić administratora kolejek (ang. *proces queue*).

Metoda szeregowania zadań

W systemie QNX każdy z procesów ma przyporządkowany priorytet w zakresie od 0 (najniższy) do 30 (najwyższy). Stosowana jest wywłaszczająca strategia szeregowania. W ramach jednego priorytetu są możliwe trzy algorytmy szeregowania: karuzelowy (*Round Robin*), FIFO i adaptacyjny.

W algorytmie karuzelowym proces zwalnia procesor, gdy upłynie przydzielony mu kwant czasu (ang. *timeslice*) lub zostanie on wywłaszczony przez proces o wyższym priorytecie. W algorytmie FIFO proces zwalnia procesor, gdy wykona wywołanie blokujące jego wykonanie lub zostanie wywłaszczony przez proces o wyższym priorytecie. Algorytm adaptacyjny nie jest stosowany w systemach czasu rzeczywistego, a raczej w systemach konwersacyjnych. Do wyboru mechanizmu szeregowania stosuje się funkcję **setscheduler( )**.

Jak już wspomniano, większość funkcji systemowych jest realizowana przez procesy systemowe (administratory), pracujące zwykle z wysokimi priorytetami. Z zagadnieniem tym wiąże się pewna subtelność dotycząca szeregowania. Gdy proces o niskim priorytecie żąda dłuższej usługi administratora o wysokim priorytecie, to uzyskuje on niejako priorytet administratora, blokując tym samym dostęp do procesora innym procesom o wyższym priorytecie niż proces klienta. Problem ten jest rozwiązywany w bardzo interesujący sposób. Przez użycie funkcji **qnx_pflags( )** z parametrem *_PPF_PRIORITY_FLOAT* modyfikuje się metodę szeregowania tak, aby proces wykonujący funkcję **Receive( )** (serwer) uzyskiwał czasowo priorytet procesu klienta, który wysłał komunikat, co pozwala na rozwiązanie tego problemu. Szczególnie interesująco skonstruowany jest administrator plików **Fsys.** Jest to proces wielowątkowy (ang. *multithread*). Gdy odbiera on żądanie od procesu klienta pracującego z określonym priorytetem, tworzy wątek obsługujący klienta i nadaje mu jego priorytet (ang. *client driven priority*).

Obsługa przerwań

W systemie QNX obsługa przerwań została zaprojektowana bardzo starannie. Ze względu na bezpieczeństwo systemu, programista nie ma bezpośredniego dostępu do kontrolera przerwań, który jest nadzorowany przez mikrojądro. Procedury obsługi przerwań są pisane jako procedury w języku C w kontekście określonego procesu i mają dostęp do jego zmiennych globalnych. Procedura **qnx_hint_attach** (*unsigned intnum, void_far *handler, unsigned ds*) włącza procedurę *handler* w łańcuch procedur obsługi przerwania *intnum* (jedno przerwanie może aktywować wiele procedur). Procedura *handler* jest dostarczana przez programistę. Istnieją dwa możliwe schematy użycia procedury obsługi przerwania:

- gdy wymagana jest krótka akcja, jak np. odbiór znaku z terminala, działanie jest wykonywane w całości w procedurze obsługi przerwania;
- gdy przerwanie sygnalizuje zdarzenie wymagające dłuższej akcji, procedura obsługi przerwania generuje proxy, która odblokowuje proces wykonujący tę akcję i podlegający zwykłym procedurom szeregowania.

Procedury obsługi przerwań są wielopoziomowe. Oznacza to, że przerwanie o wyższym priorytecie może przerwać procedurę obsługi przerwania o niższym priorytecie. Procedury obsługi przerwań mogą być tworzone tylko przez programistę posiadającego status administratora systemu (supervisora).

Obsługa czasu

System QNX ma rozbudowany zestaw usług przeznaczonych do obsługi czasu. W szczególności możliwe jest tworzenie

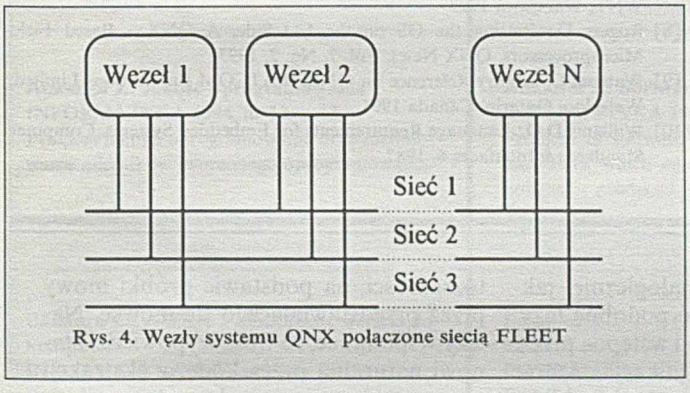
timerów (funkcja **mktimer( )**), ich ustawianie (funkcje **abstimer( )** i **reltimer( )**) oraz kasowanie (funkcja **rmtimer( )**). Ustawiony timer może zawiesić proces na określony czas i(lub) po jego upływie wygenerować sygnał albo proxy. Usługi te są wykorzystywane do implementacji przeterminowań.

Obsługa wyjątków

System QNX wykrywa zdarzenia awaryjne oraz nieprawidłowe działanie procesu i generuje w takich przypadkach sygnały. Przykładowo, sygnał może być generowany przy próbie dostępu do niedozwolonego obszaru pamięci (*SIGSEGV*) lub błędzie parzystości pamięci (*SIGBUS*). Sygnały są generowane przez system operacyjny lub proces. Mechanizm sygnałów pochodzi z systemu UNIX i szczegółowo jest opisany w [2, 9]. Proces ma możliwość obsługi sygnału i tym samym przeciwdziałania skutkom awarii. Oznacza to, że gdy wystąpi sygnał **sig**, proces może wykonać dostarczoną przez programistę funkcję **func**. Procedurę obsługi sygnału **sig** instaluje się przez funkcję **signal**(*int* **sig, void**(***func**)(*int*)). Sygnał **sig**, może być generowany także przez inny proces przez wykonanie funkcji **raise**(*int sig*).

PRACA W SYSTEMACH ROZPROSZONYCH

System QNX jest z natury systemem rozproszonym. Składa się z węzłów połączonych siecią lokalną FLEET (*Fault-tolerant, Load-balancing, Efficient, Extensible, Transparent*), co pokazano na rys. 4. System obsługuje jednocześnie do trzech sieci, co zwiększa niezawodność połączenia i przepustowość. W przypadku awarii jednej z sieci, jej funkcje przejmują automatycznie pozostałe.



Rys. 4. Węzły systemu QNX połączone siecią FLEET

Sieć jest medium, przez które są przesyłane komunikaty. Proces wykonywany w określonym węźle może korzystać z usług administratorów wykonywanych w innych węzłach. Tym samy z każdego węzła są dostępne zasoby innych węzłów, jak system plików, zainstalowane urządzenia oraz procesy-serwery. Dołączenie kolejnego węzła do sieci dodaje jego zasoby do ogólnej puli zasobów systemu. W chwili obecnej są dostępne drivery sieci Ethernet, Token, Ring, Arcnet, a także moduł TCP/IP, który umożliwia integrację QNX-a z siecią Internet.

Interfejs z użytkownikiem

Producent dostarcza kilka systemów graficznej komunikacji z użytkownikiem. Podstawowym interfejsem jest QNX Windows. Jest to łatwy w użyciu i szybki system prezentacyjny. Dla QNX Windows jest dostępny program IEDIT ułatwiający tworzenie aplikacji graficznych. Od wersji 4.2 jest oferowany standardowy interfejs X-Window w implementacji Motif. Dla sterowników wbudowanych jest dostępna okrojona wersja o nazwie Photon, wymagająca tylko 256 KB pamięci.

Integracja ze środowiskiem wykonawczym

Ponieważ QNX jest systemem modularnym, zbudowanym z mikrojądra i niezależnych administratorów, możliwe jest wygenerowanie systemu dla sterownika wbudowanego. W takim przypadku kod zawiera mikrojądro, niezbędne administratory (np. Proc, Dev), bibliotekę Slib oraz proces aplikacyjny, umieszczone w pamięci ROM. Przykład zastosowania QNX-a w sterownikach wbudowanych podano w pracy [8]. Producent dostarcza pakiet o nazwie *Data Acquisition Manager*, obsługujący typowe przemysłowe karty interfejsowe. Tworzenie systemowych procesów obsługi urządzeń (ang. *device drivers*) jest udokumentowane i stosunkowo proste.

Narzędzia wspomagające

Podstawowym narzędziem tworzenia aplikacji w systemie QNX jest język C. Komplikatorem systemowym jest WATCOM C spełniający wymagania specyfikacji ANSI X3.159. 1989. Wersja 9.5 może generować kod 16- lub 32-bitowy, w tym kod zoptymalizowany dla procesora Pentium. Dostarczany jest także debugger symboliczny WVIDEO, analizator wydajności oraz system administrowania projektem RCS (*Revision Control System*). Debugger może pracować na innym węźle niż uruchamiany program. Od niedawna dostępny jest kompilator WATCOM C++.

Charakterystyki czasowe systemu

Poniżej zostaną podane ilościowe miary wydajności systemu i porównane z innymi systemami podobnej klasy (LynxOS/386, VENIX/386). Za podstawowe charakterystyki czasowe, będące miarą jakości systemów czasu rzeczywistego, uważa się:

- czas przełączenia zadań (ang. *context switching time*) – czas potrzebny na zachowanie stanu zadania *A* i wznowienie zadania *B*,
- czas aktywacji przerwania (ang. *interrupt latency*) – czas między wystąpieniem przerwania a wykonaniem pierwszej instrukcji procedury jego obsługi,
- czas szeregowania (ang. *scheduling latency*) – czas wybrania i wznowienia wybranego zadania.

Tabela 1. Charakterystyki czasowe systemów

	QNX4	LynxOS/386	VENIX/386
Czas przełączenia zadań	26 μs	13 μs	300 μs
Czas aktywacji przerwania	11 μs	13 μs	50 μs
Czas szeregowania	27 μs	41 μs	brak danych

W tabeli 1 podano charakterystyki czasowe systemów QNX 4.11, LynxOS/386 i VENIX/386 dla testów wykonanych na

komputerze PC386, 33 MHz. Dane pochodzą z materiałów informacyjnych producentów.

* * *

System QNX jest od podstaw zaprojektowany jako system czasu rzeczywistego do zastosowań krytycznych. Jest dobrze przetestowany o czym świadczy zastosowanie go do sterowania elektrowniami atomowymi [3] i ruchem pociągów w tunelu pod kanałem La Manche. Jest to system w dużym stopniu skalowalny. Może być zainstalowany zarówno w stosunkowo niewielkich sterownikach wbudowanych, jak i tworzyć systemy złożone z wielu węzłów połączonych siecią. Jest dobrze dostosowany do z natury zdecentralizowanych systemów sterowania i systemów telekomunikacyjnych. W przypadku, gdy aktualna instalacja ma zbyt małą wydajność, można bez trudności dołączyć kolejny węzeł. QNX nadaje się dobrze do tworzenia zwielokrotnionych sprzętowo systemów o dużej niezawodności, zbudowanych z dostępnych powszechnie komponentów, a przez to stosunkowo tanich [4]. Pod względem charakterystyk czasowych system nie ustępuje najlepszym systemom dostępnym aktualnie na rynku (LynxOS). Aplikacje są tworzone w języku C (ewentualnie w C++). Z jednej strony jest to zaletą, gdyż jest to jeden z najpowszechniej stosowanych obecnie języków. Jednak język ten nie wymusza bezpiecznego programowania, a programy mogą być mało czytelne i trudne do modyfikacji. Ponieważ system spełnia normę POSIX, większość oprogramowania źródłowego systemu UNIX daje się przenieść na QNX-a bez większych trudności. Do słabszych stron systemu należy zaliczyć ograniczoną liczbę platform sprzętowych, na których jest dostępny oraz generalnie gorszą zgodność z UNIX-em niż systemy konkurencyjne. Zarówno LynxOS, jak i VENIX wywodzą się bezpośrednio z AT & T UNIX System V i są z nim zgodne na poziomie kodu binarnego. Ze względu na dużą liczbę instalacji systemu QNX należy spodziewać się, że jego rozwój będzie kontynuowany. Najistotniejsze informacje porównawcze zestawiono w tabeli 2.

Tabela 2. Podstawowe cechy systemów QNX4, LynxOS i VENIX

	QNX4	LynxOS	VENIX/386
Platforma sprzętowa	PC 386, 486, Pentium	PC 386, 484 Motorola 6803, 68040 MIPS R3000, R4000 HP 7100	PC 386, 486
Pamięć wirtualna	nie	tak	tak
Przetwarzanie rozproszone	tak	tak	ograniczone
Języki tworzenia aplikacji	C, C++	C, Ada	C, C++, Ada
Komunikacja międzyzadaniowa	komunikaty, proxy	jak w UNIX System V	jak w UNIX system V
Typ jądra	modularne	monolityczne	monolityczne
Interfejs z użytkownikiem	QNX Windows, OSF/Motif, Photon	OSF/Motif	OSF/Motif
Zapotrzebowanie na zasoby	małe	duże	duże

LITERATURA

[1] Bach M.J.: The Design of the UNIX Operating System. Prentice Hall, New Jersey 1986

[2] Hildebrand D.: An Architectural Overview of QNX. QNX Software Systems Ltd. Kanata 1993

[3] Hildebrand D.: A Microkernel POSIX OS for Realtime Embedded Systems. Embedded Computer Conference, Santa Clara, California, April, 1993

[4] Irribaren R.: A Voice Processing System That Never Quits. QNX News, Vol. 7, No. 1, 1993

[5] QNX Reference Guide, Version 2.1. Quantum Software Systems, Kanata 1990

[6] QNX 4.1 Operating System – System Architecture. Quantum Software Systems, Kanata 1992

[7] Rochkind M.J.: Programowanie w systemie UNIX dla zaawansowanych. WNT, Warszawa 1993

[8] Rogers D.: Putting the OS on the I/O Side, A QNX – Based Field Microprocessors. QNX News, Vol. 7, No. 2, 1993

[9] Watcom C, Library Reference for QNX. WATCOM Publications Limited, Waterloo, Ontario, Canada 1991

[10] Williams D.O.: Language Requirements for Embedded Systems. Computer Standards & Interfaces 6, 1987.

AWEK akustyczne wejście do komputera

dokończenie ze s. 21

W procesie ARG, analogicznie jak w ARM, można wyróżnić podobne fazy operacyjne, a więc: źródło i wstępne przetwarzanie, pomiar i klasyfikacja. Automatyczne rozpoznawanie głosów ARG obejmuje głównie dwa różniące się zadania: identyfikację oraz weryfikację głosów.

Identyfikacja głosów, to procedura przypisania nieznanej wypowiedzi do mówcy z danego zbioru mówców. Inaczej mówiąc, w populacji M mówców należy wskazać mówcę, którego cechy osobnicze głosu są najbardziej zbliżone do cech głosu zawartych w wypowiedzi nieznanego mówcy. Weryfikacja głosów AWG, będąca częścią opracowywanego wejścia akustycznego AWEK, jest procedurą potwierdzenia lub odrzucenia zgłoszonej tożsamości, na podstawie próbki mowy przez przedstawiającego się mówcę. Należy wspomnieć, że treść wypowiedzi stanowi naturalną przeszkodę w ekstrakcji parametrów opisujących cechy osobnicze. Z tego względu stosuje się określone modele analizy sygnału mowy, które neutralizują wpływ tego *co się mówi* na to *kto mówi*.

Celem projektu „Akustyczne wejście do komputera" jest opracowanie praktycznych podstaw i modelu systemu automatycznego rozpoznawania mowy o słowniku do kilkuset wyrazów wraz z weryfikacją głosu mówcy (operatora). Oznacza to opracowanie systemu pozwalającego na wprowadzenie informacji i komend głosem operatora, którego tożsamość może być również zweryfikowana.

INFO '94 FESTIWAL
Kraków, 8-10 listopada 1994 r.

Informatyka w nauce polskiej

Najważniejszą imprezą tegorocznego Info-Festiwalu (Kraków 8–10 listopada 1994 r.) była ogólnopolska konferencja naukowa pn. „Zaawansowane technologie informatyczne i telekomunikacyjne w informatyce polskiej", zorganizowana pod patronatem przewodniczącego KBN – prof. Witolda Karczewskiego. Przedstawiono na niej m.in. strategię rozwoju infrastruktury informatycznej w nauce polskiej, omówiono wykorzystanie komputerów dużej mocy w ośrodkach akademickich i wykorzystywanie programów typu CAD/CAM/CAE w nauce. Duża część seminarium poświęcona była telekomunikacji, sieciom rozległym oraz poczcie elektronicznej. Konferencja była w pełni zintegrowana z ekspozycją „Informatyka dla nauki", w której prezentowano rozwiązania firm ATM, CONTROL-SOFT, IBM, NASK, SOLIDEX, SUN MICROSYSTEMS. Czynna też była wystawa podzielona na kilka działów tematycznych: Windows World, Grafika i wizualizacja komputerowa, Multimedia, Telekomunikacja, Ranking sprzętu komputerowego i kas fiskalnych. Ciekawostką dla zwiedzających wystawę były magnetyczne karty wstępu.

Januchta M., Królikowski Z.: POSTGRES jako narzędzie budowy aktywnych i dedukcyjnych systemów baz danych
INFORMATYKA 1994, nr 12, s. 4
Zastosowanie systemu zarządzania bazą danych POSTGRES umożliwiającego bardziej efektywne niż w przypadku podobnych produktów komercyjnych tworzenie aktywnych i dedukcyjnych systemów baz danych.

Januchta M., Królikowski Z.: The POSTGRES system as a tool for building of active and deductive database systems
INFORMATYKA 1994, No. 12, p. 4
Application of the POSTGRES database management system, which enables more effective as in similar commercial products building of active and deductive database systems.

Januchta M., Królikowski Z.: POSTGRES-System als Hilfsmittel für den Bau von aktiven und didaktiven Datenbanksystemen
INFORMATYKA 1994, Nr. 12, S. 4
Anwendung vom POSTGRES-Datenbankverwaltungssystem, das den mehr effektiven als in ähnlichen kommerziellen Produkten Bau von aktiven und deduktiven Datenbanksystemen ermöglicht.

Kwaśnicka H.: Próba naśladowania natury – algorytmy genetyczne w informatyce
INFORMATYKA 1994, nr 12, s. 12
Charakterystyka istoty, schemat działania, sposób opisu oraz obszary zastosowań algorytmów genetycznych jako nowego narzędzia informatyki.

Kwaśnicka H.: An attempt of nature simulating – the genetic algorithms in computer science
INFORMATYKA 1994, No. 12, p. 12
Characteristics of the idea, function schema, description method and application areas of genetic algorithms as a new computer science tool.

Kwaśnicka H.: Ein Versuch der Naturnachahmung – die genetischen Algorithmen in Informatik
INFORMATYKA 1994, Nr. 12, S. 12
Eine Charakteristik vom Begriff, Wirkungsschema, Beschreibungsmethode und Anwendungenbereiche der genetischen Algorithmen als neues Hilfsmittel der Informatik.

Auksztol J.: Jakość oprogramowania
INFORMATYKA 1994, nr 12, s. 17
Problem jakości oprogramowania oraz zdefiniowanie mierników tworzonego oprogramowania.

Auksztol J.: Software quality
INFORMATYKA 1994, No. 12, p. 17
Problem of software quality and definition of measurers, which enable effective statement of bought or produced program product's quality.

Auksztol J.: Softwarequalität
INFORMATYKA 1994, Nr. 12, S. 17
Das Problem der Softwarequalität und Definieren der Messer zur objektiven Qualitätsbestimmung der angekauften oder hergestellten Softwareprodukten.

Basztura C.: AWEK – akustyczne wejście do komputera
INFORMATYKA 1994, nr 12, s. 20
Problem automatycznego rozpoznawania głosu oraz stan realizacji projektu akustycznego wejścia do komputera AWEK, wykonywanego w Instytucie Telekomunikacji i Akustyki Politechniki Wrocławskiej.

Basztura C.: AWEK – acoustic computer input
INFORMATYKA 1994, No. 12, p. 20
Problem of automatic voice recognition and actual realization of the AWEK – acoustic computer input, designed in Telecommunication and Acoustics Institute of Wrocław Technical University.

Basztura C.: AWEK – eine akustische Computereingabe
INFORMATYKA 1994, Nr. 12, S. 20
Das Problem der automatisierten Stimmeerkennung und Realisationsstand des AWEK-Projektes der akustischen Computereingabe, das im Institut für Telekommunikation und Akustik der Technischen Universität zu Wrocław ausgeführt wird.

Ułasiewicz J., Fryźlewicz Z.: System operacyjny QNX jako narzędzie tworzenia aplikacji czasu rzeczywistego
INFORMATYKA 1994, nr 12, s. 22
Charakterystyka struktury oraz możliwości funkcjonalnych systemu operacyjnego QNX wersja 4 do zastosowań w systemach czasu rzeczywistego oraz jego porównanie z podobnymi systemami operacyjnymi.

Ułasiewicz J., Fryźlewicz Z.: QNX operating system as a tool for real-time applications
INFORMATYKA 1994, No. 12, p. 22
Characteristics of the QNX operating system, version 4, structure and functional possibilities for real-time system applications and its comparison with similar operating systems.

Ułasiewicz J., Fryźlewicz Z.: QNX-Betriebssystem als Hilfsmittel zum Bau der Echtzeitanwendungen
INFORMATYKA 1994, Nr. 12, S. 22
Eine Charakteristik von Struktur und funktionellen Möglichkeiten des QNX-Betriebssystems, Version 4, für Anwendungen in Echtzeitsystemen, sowie sein Vergleich zu ähnlichen Betriebssystemen.

SOFTARG '94

Kolejne, VIII Międzynarodowe Targi Oprogramowania SOFTARG '94 (13–16 września br.) potwierdziły dalszy konsekwentny rozwój tej specjalistycznej, ukierunkowanej na oprogramowanie aplikacyjne i narzędziowe imprezy informatycznej. Określenie „międzynarodowe" nadano przed trzema laty SOFTARGOM trochę na wyrost, ponieważ zagraniczni producenci i dystrybutorzy oprogramowania nadal nie zdradzają większego zainteresowania tą imprezą, świadomi faktu, że w tym sektorze rynku nie mają zbyt dużych szans na osiągnięcie sukcesu. Natomiast dla krajowych producentów oprogramowania SOFTARG był i pozostaje nadal najważniejszym miejscem dorocznej prezentacji osiągnięć w tej kategorii produktów, a organizowane od samego początku konkursy na najlepszy produkt krajowy – niezwykle skutecznym i atrakcyjnym sposobem promocji wyrobów.

Należy przypomnieć, że podstawowym celem wspomnianych konkursów jest wyłanianie produktów najlepszych, przy czym o ocenie decydują m.in. takie kryteria, jak liczba wdrożeń, jakość dokumentacji oraz znaczenie gospodarcze lub społeczne produktu. Od kilku lat organizatorem konkursów jest Polskie Towarzystwo Informatyczne (PTI), co gwarantuje wysoki poziom merytoryczny oraz obiektywizm oceny produktów.

Oczywiście SOFTARGI służą nie tylko interesom producentów oprogramowania. W nie mniejszym stopniu służą one również polskim użytkownikom informatyki, którzy w jednym miejscu uzyskują możliwość bezpośredniego poznania napełniejszej spośród największych krajowych imprez informatycznych (warszawski KOMPUTER oraz poznański INFOSYSTEM) aktualnej oferty dostępnego w kraju oprogramowania aplikacyjnego i narzędziowego. Obserwacja wielkich światowych imprez informatycznych potwierdza, że punkt ciężkości zainteresowań użytkowników coraz bardziej przesuwa się właśnie na obszar oprogramowania, spychając na dalszy plan problemy wyboru sprzętu komputerowego i oprogramowania systemowego (podstawowego), które uległy daleko idącej standaryzacji.

Podobnie jak każda impreza, nie wyłączając najbardziej prestiżowych, jak np. CeBIT, również SOFTARG ma jeszcze

PIERWSZY POLSKI INTELIGENTNY
SYSTEM INFORMATYCZNY WSPOMAGAJĄCY ZARZĄDZANIE
PRODUKCJĄ - (TPPiRP), FINANSAMI - KOSZTAMI, SPRZEDAŻĄ, KADRAMI
WRAZ Z SYSTEMEM INFORMACJI KIEROWNICTWA
WSPOMAGANY SYSTEMEM SZTUCZNEJ INTELIGENCJI PC-SHELL
ORAZ EKSPERCKĄ BAZĄ WIEDZY EKONOMICZNO-FINANSOWEJ - ISAF
(DLA DUŻYCH PRZEDSIĘBIORSTW PRODUKCYJNYCH)

Perfect-Ekspert AITECH

(DZIAŁAJĄCY W OPARCIU O ŚRODOWISKO PRACY - WYŁĄCZNIE SYSTEM OPERACYJNY DOS, NetWare v3.11, v4.01 FIRMY NOVELL)

OFERUJE

PRZEDSIĘBIORSTWO PRODUKCYJNO-USŁUGOWE INNOWACJI TECHNICZNYCH

JUNISOFTEX Sp. z o.o.

ROK ZAŁOŻENIA 1987

WARTOŚĆ SPRZEDAŻY wg F-02 1992 (14,6 mld), wg F-01 na 1993.10.31 (15,4 mld)

TEL/FAX 315-131, 315-700, 321-525, 388-052, 388-054 TLX 036233

BIURO TECHNICZNO-HANDLOWE

44-100 GLIWICE ul. Kościuszki 1

WSZYSTKICH ZAINTERESOWANYCH ZAPRASZAMY NA MERYTORYCZNE ROZMOWY DO NASZEGO BIURA TECHNICZNO-HANDLOWEGO

OFERUJEMY RÓWNIEŻ INTELIGENTNE NARZĘDZIE DO PRODUKCJI EKSPERCKICH BAZ WIEDZY

PC-SHELL

ARTIFICIAL INTELLIGENCE

DLA JEDNOSTEK BUDŻETOWYCH (NAUKA, OŚWIATA, URZĘDY MIAST I GMIN, SŁUŻBA ZDROWIA) BEZ PRAWA DO KOMERCYJNEGO WYKORZYSTYWANIA W CENIE 5.000.000,- (BEZ VAT) - INFORMACJE I ZAMÓWIENIA PROSIMY SKŁADAĆ BEZPOŚREDNIO U PRODUCENTA ARTIFICIAL INTELLIGENCE LABORATORY - AITECH, KATOWICE 14 SKR. POCZT. 2804.

DLA POZOSTAŁYCH JEDNOSTEK GOSPODARCZYCH BEZ PRAWA KOMERCYJNEGO WYKORZYSTYWANIA TJ. WYŁĄCZNIE NA WŁASNE POTRZEBY DO KOŃCA I KW. 1994 ROKU W CENIE PROMOCYJNEJ 10.000.000,- (BEZ VAT), DLA KUPUJĄCYCH DO CELÓW KOMERCYJNYCH DO KOŃCA I KW. 1994 25.000.000,- (BEZ VAT), LICENCJA OGRANICZONA NA NIEOKREŚLONĄ ILOŚĆ ODBIORCÓW KOŃCOWYCH.

POSZUKUJEMY DO WSPÓŁPRACY LUB NAWIĄŻEMY WSPÓŁPRACĘ Z EKSPERTAMI Z DOWOLNYCH DZIEDZIN NAUKI I TECHNIKI.

0/3/93

wiele słabych punktów. Są one sukcesywnie eliminowane, ale spełnienie wszystkich postulatów nie jest możliwe, choćby ze względu na często występującą sprzeczność interesów różnych zainteresowanych stron, np. wystawcy domagają się skrócenia czasu i godzin otwarcia, natomiast zwiedzający – ich wydłużenia.

Jako jeden z członków Jury Konkursu SOFTARG '94 chciałbym podkreślić, że konkursy tej imprezy przez wywołanie ostrego współzawodnictwa wniosły i nadal wnoszą istotny wkład w podnoszenie merytorycznego poziomu krajowych produktów programowych, a zwłaszcza w poprawę ich walorów użytkowych (coraz lepsza dokumentacja, nie ustępująca pod względem staranności opracowania oraz formy edytorskiej najbardziej renomowanym produktom zagranicznym). Przy okazji chciałbym również odeprzeć podniesiony zarówno w prasie codziennej, jak i specjalistycznej, zarzut nadmiernego „rozwodnienia" przyznanych w br. nagród. Otóż łączna liczba nagród i wyróżnień utrzymuje się w ostatnich trzech latach na zbliżonym poziomie:

- 1992 – 13 nagród i 12 wyróżnień (łącznie 25 na 65 zgłoszeń),
- 1993 – 11 nagród i 12 wyróżnień (łącznie 23 na 96 zgłoszeń),
- 1994 – 12 nagród i 12 wyróżnień (łącznie 24 na 74 zgłoszeń).

Znaczna liczba przyznawanych nagród i wyróżnień jest odbiciem wzrastającego poziomu wyrobów.

Z obowiązku kronikarskiego podaję kompletną listę przyznanych nagród i wyróżnień SOFTARG '94. Wydaje się, że przy ich ogłaszaniu należałoby od przyszłego roku wprowadzić modyfikację, polegającą na wymienianiu (wzorem innych dziedzin twórczości) również nazwisk autorów oprogramowania, a co najmniej kierowników zespołów realizatorskich. Stanowiłoby to niezbędny wyraz uznania dla obecnie anonimowych autorów sukcesu firm.

Jury Konkursu pod przewodnictwem prof. dr. Andrzeja Grzywacza przyznało nagrody i wyróżnienia następujący firmom:

Nagrody I stopnia

- PROLOC z Katowic – za oprogramowanie rozproszonych systemów sterowania działających w sieciach przemysłowych.
- MAX ELEKTRONIK z Zielonej Góry – za kompleksowy system zarządzania biblioteką „PROLIB".

Nagrody II stopnia

- YOUNG DIGITAL POLAND z Gdańska – za elementarz do nauki języka angielskiego.
- VULCAN z Wrocławia i Wydawnictwa Szkolne i Pedagogiczne z Warszawy – za programy edukacyjne dla szkół.
- TECHNO-PROGRESS z Krakowa – za komputerowy system informacji przestrzennej „Net-GRAF".
- „RoboBAT" z Krakowa – za integrowane oprogramowanie analizy konstrukcji inżynierskich.

Nagrody III stopnia

- BORSA + POREMBSKI z Gliwic – za system informacji geograficznej.
- PTH TECHMEX z Bielska Białej – za system finansowo-księgowy „Premiera".
- JUNISOFTEX z Gliwic – za system zarządzania przedsiębiorstwem.
- PROKOM z Gdyni – za zintegrowany system zarządzania.
- CSBI z Warszawy – za kompleksowy system zarządzania „PRO/MIS".
- MacroSoft z Warszawy – za pakiet programowy „MacroBASE dla systemu UNIX".

Wyróżnienia:

- POTRONICS z Łodzi – za zintegrowany pakiet programowy obejmujący edytor tekstu, arkusz kalkulacyjny i moduł wykresów.
- ELBA z Wrocławia – za system raportowania dla banku.
- AITECH z Katowic – za szkieletowy system ekspercki „PC-Shell".
- DIGITLAND z Krakowa – za system zarządzania firmą „HURT".
- AutoR z Radomia – za system wspomagania projektowania w mechanice.
- KORELACJA z Krakowa – za program „POMOC SPOŁECZNA dla Gminnych i Miejskich Ośrodków Pomocy Społecznej".
- INTEGRAL z Gliwic – za system wspomagania pracy rejonu energetycznego.
- TARAN z Mielca – Za system zarządzania przedsiębiorstwem komunikacyjnym.
- REGEN-DRUK z Gliwic – za system „Od pomiarów do fakturowania".
- LOGOTEC z Mysłowic – za system zarządzania danymi produkcyjnymi.
- ALTKOM MATRIX z Warszawy – za system finansowo-księgowy „SYMFONIA".
- CENTRALNEMU OŚRODKOWI INFORMATYKI GÓRNICTWA z Katowic – za komputerowy system dla administracji terenowej.

Ponadto Jury zaproponowało przyznanie następujących nagród specjalnych:

- Dyplomy Wojewody Katowickiego dla firm najbardziej zaawansowanych we wdrażaniu systemu dla administracji rządowej i samorządowej: REKORD z Bielska, KORELACJA z Krakowa i MICOMP z Katowic.
- Nagrodę Prezesa Międzynarodowych Targów Katowickich w postaci przyznania stoiska na SOFTARG '95 za program dydaktyczny „SUPERMEMO" firmie SUPERMEMO WORLD z Poznania.
- Nagrodę Dyrektora Śląskiej Dyrekcji Okręgowej PKP 5 mln zł – za rozwój oprogramowania wspomagającego proces przewozowy, firmie INWAR z Sieradza.
- Nagrodę PTI – Oddziału Górnośląskiego za konsekwentny rozwój własnego narzędziowego oprogramowania „MacroBASE", firmie „MacroSoft" z Warszawy.
- Nagrodę Prezydenta Miasta Katowic 5 mln zł – twórcom programu edukacyjnego dla szkół, firmie VULCAN z Wrocławia.

Ze środków przyznanych przez Bank Śląski S.A. proponuje się przyznać nagrody finansowe autorom następujących programów:

- Program raportowania sytuacji bankowej – firma „ELBA" z Wrocławia – 15 mln zł;
- Oprogramowanie rozproszonych systemów sterowania – firma PROLOC z Katowic – 10 mln zł;
- System zarządzania biblioteką – firma MAX ELEKTRONIK z Zielonej Góry – 10 mln zł;
- Elementarz języka angielskiego – firma YOUNG DIGITAL z Gdańska – 5 mln zł;
- Komputerowy system informacji przestrzennej „Net-Graf" firma TECHNO-PROGRESS z Krakowa – 5 mln zł;
- Zintegrowane oprogramowanie analizy konstrukcji inżynierskich firma „RoboBAT" z Krakowa – 5 mln zł.

WŁADYSŁAW KLEPACZ

Cena 45 000 zł

FoxPro 2.6 dla Unix-a, Macintosha i Power Macintosha

Microsoft rozpoczął produkcję najnowszej wersji systemu zarządzania relacyjnymi bazami danych (SZRBD) FoxPro 2.6. System FoxPro 2.6 dla Unixa jest jedyną bazą danych działającą na czterech platformach systemowych: Macintosh, Windows, MS-DOS i obecnie Unix w postaci SCO Unix i XENIX. FoxPro 2.6 oferuje kompletną zgodność między platformami. Każdy z programów może korzystać z danych zawartych na innych platformach; także aplikacje można bez żadnych zmian uruchamiać na innych platformach. FoxPro 2.6 dla Unixa dostarcza programiście pełnego zestawu narzędzi, niedostępnych ani w FoxBASE+, ani w dBASE IV 2.0 dla Unixa. Może pracować z wieloma terminalami jednocześnie, dopuszcza też użycie wielu różnych typów terminali, co umożliwia tworzenie tanich aplikacji pod klucz.

FoxPro 2.6 dla Macintosha i Power Macintosha wykorzystuje efektywne możliwości systemu operacyjnego Macintosha System 7 i technologii Power Macintosh. Jego cechy to potężny graficzny Catalog Manager służący do zarządzania tabelami, zapytaniami, formularzami, raportami i innymi elementami baz danych. Zawiera też nowe elementy Wizard (Creator) ułatwiające wykonywanie często powtarzanych prac oraz ponad 50 nowych rozszerzeń języka. Dostępna jest także wersja Microsoft FoxPro 2.6 Professional dla Macintosha.

System 9-1-1

Ponad 22% wezwań do nagłych wypadków przechodzi w Ameryce przez telefony komórkowe, które – jeśli wzywający nie wie, gdzie się znajduje – bardzo trudno jest zlokalizować. W Huston (Texas) instalowany jest system Przyjmowania Nagłych Wezwań (9-1-1) zdolny do śledzenia rozmów w sieci telefonii komórkowej. System ten jest oparty na centralach Northom Telecom oraz serwerach firmy Digital – Alfa AXP i komputerach osobistych tej firmy zintegrowanych z telefonią – Computer Integrated Telephony (CIT). System będzie mógł zidentyfikować komórki sieci, które wysłały wezwanie, automatycznie zlokalizować wzywającego i przekazać wezwanie do waściwych agend (policji, straży pożarnej, pomocy medycznej).

IBM PC Serwer

IBM rozpoczął produkcję nowego serwera o nazwie IBM PC Serwer, który został zaprojektowany specjalnie dla potrzeb małych firm wykorzystujących sieci złożone z komputerów osobistych. Serwer charakteryzuje się dużą elastycznością dzięki połączeniu mikroprocesorów 486DX2/66 MHz i Pentium 60 MHz z nową architekturą magistrali PCI (Peripherial Component Interconnect). Sterownik IDE (Intelligent Drive Electronics), PCI i nowy adapter IBM SCSI-2 Fast PCI umożliwiają dalszą modernizację systemu. PC Serwer jest zaopatrzony także w dziewięć gniazd rozszerzeniowych, z których jedno przenaczone jest dla obu standardów: PCI i EISA (Extended Industry Standard Architecture). Pozwoli to na stopniowe przechodzenie do technologii PCI w miarę, jak na rynku będą pojawiać się nowe adaptery i aplikacje. PC Serwer został wyposażony w nowy mechanizm chłodzący – „Flothru" – zapewniający dodatkową cyrkulację powietrza i przedłużający czas życia elektronicznych podzespołów serwera. IBM daje na ten serwer trzyletnią gwarancję.

LAN PCI z SMC

Firma SMC (Standard Microsystems Corporation) wprowadziła w październiku br. na rynek adapter sieciowy Ethernet współpracujący z szyną PCI (Peripheral Component Inferface). SMC EtherPower jest szybkim adapterem sieciowym, który może przejmować szyny w celu zmaksymalizowania wydajności. Wraz z adapterem dostarczane są różne sterowniki dostosowane do popularnych systemów operacyjnych. Właściwości typu Plug and Play szyny PCI sprawiają, że instalacja adaptera jest bardzo prosta.

Oto cechy EtherPower: pełna konfigurowalność przy załączeniu zasilania, szybkość przesyłania do 123 MB/s po przejęciu szyny, w pełni dupleksowy Ethernet, dostępny w wersjach dla skrętki (RJ-45) i kabla koncentrycznego (BNC), sterowniki dla Novell NetWare, Windows for Workgroups, Windows NT, SCO Unix i in. Adapter EtherPower został przetestowany na zgodność z szeroką gamą komputerów z procesorami 486 i Pentium, hubami i różnymi systemami sieciowymi. Pod koniec br. pojawi się dwukanałowa wersja adaptera. W Polsce dystrybutorem firmy SMC jest DHI Computer 2000.

Żegnaj DHI, witaj C2000

Dom Handlowy Informatyki, jeden z największych dystrybutorów sprzętu i oprogramowania w Polsce (w latach 1993–1994 obroty wyniosły 700 mld zł), od października 1994 r. zmienił nazwą na Computer 2000 Polska. DHI związał się we wrześniu 1992 r. z Computer 2000 (C2000) – największym europejskim dystrybutorem sprzętu komputerowego, oprogramowania, urządzeń peryferyjnych oraz produktów telekomunikacyjnych. C2000 ma 35 filii w 23 krajach. Decyzja DHI o zmianie nazwy jest podyktowana, jak stwierdził przedstawiciel zarządu firmy, chęcią zaznaczenia obecności europejskiego giganta w Polsce. Najpóźniej w ciągu roku spodziewane jest wejście na nasz rynek dwóch światowych gigantów – amerykańskich firm Marisel i Ingram Micro. Jak dotąd C2000 z 2,3 mld USD obrotów rocznie przoduje na europejskim rynku. Marisel ma obroty wynoszące 5,1 mld USD, w Europie wynoszą one 1 mld USD, obroty Ingram Micro wynoszą 5,5 mld USD, z czego w Europie też 1 mld USD. Rynek polski, podobnie jak rynki innych krajów tej części Europy, jest bardzo interesujący dla światowych dystrybutorów. Wejście gigantów na naszy rynek – jak pokazują doświadczenia innych krajów, gdzie to już się stało – może obniżyć ceny w tak znaczący sposób, że mali dystrybutorzy zostaną wyeliminowani. Przygotowanie do wejścia „smoków" jest chyba też przyczyną dokonanej we wrześniu 1994 r. fuzji MSP (obroty w III pierwszych kw. 1994 r. wyniosły ok. 10 mln USD) z czeskim TH'system, którego obrót wyniósł w 1994 r. 28 mln USD.

Opracowała
KRYSTYNA
KARWICKA-RYCHLEWICZ

Odporność na błędy dla systemu NetWare 4

Novell ogłosił, iż wszystkim użytkownikom NetWare 4 dostarcza także technologię odporności na błędy NetWare System Fault Tolerance (SFT). System NetWare 4.1 SFT III jest oparty na NetWare SFT III 3.11. NetWare 4.1 SFT III jest dostępny jednocześnie z NetWare 4.1 jako osobny moduł funkcjonalny, ściśle zintegrowany z systemem operacyjnym NetWare 4. Zintegrowana w NetWare SFT III technologia lustrzanego serwera zapobiega skutkom awarii serwera, dzięki podtrzymywaniu działania dwóch serwerów o identycznej zawartości pamięci operacyjnej i dysków. Kiedy jeden serwer ulegnie uszkodzeniu, drugi przejmuje jego funkcje bez straty danych i przerywania pracy sieci. System NetWare 4.1 SFT III pozwala na korzystanie z dwóch tanich serwerów PC. Ma wiele nowych cech funkcjonalnych, m.in. jest ściśle zintegrowany z systemem operacyjnym NetWare 4 i ma wszystkie jego właściwości. Dzięki protokołom komunikacyjnym Apple Talk i TCP/IP umożliwia dołączanie klientów Macintosha i Unixa, jest polepszona w nim usługa drukowania, informuje administratora systemu lub grupy administratorów o awarii sprzętu i przejściu na pracę z zapasowego serwera. Zawiera wsparcie dla programów zarządzania archiwizacją i dla systemu zarządzania NMS. Wszyscy obecni i nowi użytkownicy NetWare SFT III, którzy zarejestrują produkt do 31 grudnia 1994 r., otrzymają pełną aktualizację przed końcem marca 1995 r.